BİNEK ARAÇ BAGAJ HACMİNİN HESAPLANMASI İÇİN BİR ALGORİTMA GELİŞTİRİLMESİ

İBRAHİM HALİL DEMİR

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİNEK ARAÇ BAGAJ HACMİNİN HESAPLANMASI İÇİN BİR ALGORİTMA GELİŞTİRİLMESİ

İbrahim Halil DEMİR

Prof. Dr. Necmettin KAYA
(Danışman)

YÜKSEK LİSANS TEZİ
MAKİNE MÜHENDİSLİĞİ ANABİLİM DALI

BURSA– 2019

TEZ ONAYI

İbrahim Halil DEMİR tarafından hazırlanan "Binek Araç Bagaj Hacminin Hesaplanması İçin Bir Algoritma Geliştirilmesi" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Makine Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Necmettin KAYA

Başkan :	Prof. Dr. Necmettin KAYA Uludağ Üniversitesi Mühendislik Fakültesi Makine Mühendisliği Anabilim Dalı	İmza
Üye :	Dr. Öğr. Üyesi Erol SOLMAZ Uludağ Üniversitesi Mühendislik Fakültesi Otomotiv Mühendisliği Anabilim Dalı	İmza
Üye :	Dr. Öğr. Üyesi Celalettin YÜCE Bursa Teknik Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi Mekatronik Mühendisliği Anabilim Dalı	İmza

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel Eren
Enstitü Müdürü
19/4/2019

U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

19/06/2019

İbrahim Halil DEMİR

ÖZET

Yüksek Lisans Tezi

BİNEK ARAÇ BAGAJ HACMİNİN HESAPLANMASI İÇİN BİR ALGORİTMA GELİŞTİRİLMESİ

İbrahim Halil DEMİR

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Makine Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Necmettin KAYA

Bu tez çalışmasında, binek ve ticari araçların bagaj hacimlerinin uluslararası standartlara göre hesaplanabilmesi için bilgisayar destekli tasarım (CAD) yazılımı içerisinde çalışan bir yerleşim algoritması geliştirilmiş, algoritma kodlanarak örnek modeller üzerinde uygulanmıştır. Dış yüzeyleri ve iç yüzeyleri verilen bagaj geometrik modeli üzerinde, ISO 3832 ve DIN 70020 standartlarında yer alan yöntem referans alınarak bagaj içine konulan hacimsel blokların dik yerleşim düzenlerine göre dizilmesi geliştirilen algoritma ile gerçekleştirilmiş, maksimum sayıda bloğun sığdığı durum bagaj hacmi olarak belirlenmiştir. Bu sayede, yeni araç tasarım sürecinde zaman ve maliyet açısından önemli kazanımlar söz konusu olacak, mevcut durumda fiziksel ortamda manuel olarak yapılan ve zaman alan testler, araç tasarım sürecinde CAD ortamında daha kısa sürede gerçekleştirilebilecektir..

Anahtar Kelimeler: Araç bagaj hacmi, DIN 70020, ISO 3832
2019, vii +68 sayfa.

ABSTRACT

MSc Thesis

DEVELOPING AN ALGORITHM TO CALCULATE VEHICLE TRUNK VOLUME

İbrahim Halil DEMİR

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Mechanical Engineering

Supervisor: Prof. Dr. Necmettin KAYA

In this thesis study, an algorithm was developed in order to calculate the trunk volumes of vehicles according to international standards. The algorithm was coded inside the CAD model and applied on the example models. Standard blocks are arranged orthogonally inside a given outer and inner surfaces of trunk volume according to ISO 3832 and DIN 70020 standards using developed algorithm. The maximum number of blocks is set as trunk volume. Therefore, there will be significant gains in terms of time and cost in the new vehicle design process. In the current situation, the manual and time-consuming tests in the physical environment can be performed in the CAD environment in the vehicle design process in a shorter time.

Key words: Trunk volume, DIN 70020, ISO 3832
2019, vii +68 pages.

TEŞEKKÜR

Yüksek lisans öğrenimim boyunca ve tez çalışmamda her daim yanımda olan, bilgisini ve tecrübesini cömertçe paylaşan, çalışmamın başarıyla sonuçlaması için göstermiş olduğu büyük emek,sabır ve desteği için saygıdeğer hocam Prof. Dr. Necmettin KAYA'ya teşekkürlerimi sunarım.

Hayatımın her döneminde yanımda olan, eğitim hayatımı her daim destekleyen, haklarını hiçbir zaman ödeyemeyeceğim sevgilimi aileme teşekkürlerimi sunarım.

İbrahim Halil DEMİR
19/06/2019

İÇİNDEKİLER

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	**Açıklama**
mm	milimetre

Kısaltmalar	**Açıklama**
CAD	Computer Aided Design (Bilgisayar Destekli Tasarım)
DIN	Deutsches Institut für Normung (Alman Standartlar Entitüsü)
ISO	International Organization for Standardization (Uluslararası Standartlar Örgütü)
SAE	The Society of Automotive Engineers (Otomotiv Mühendisleri Derneği)
BLBF	Bottom Left Back Fill (Sol Alt Köşe Doldurma)
GA	Genetik Algoritma
DGA	Diferansiyel Gelişim Algoritması
SA	Simulated Anneling (Tavlama Benzetimi)
PC	Personal Computer (Kişisel Bilgisayar)
NP	Number of Population (Popülasyon Büyüklüğü)
DG	Diferansiyel Gelişim
CR	Crossover Rate (Çaprazlama Oranı)
F	Scaling Factor (Ölçekleme Faktörü)
LAFF	Largest Area First Fit (İlk Uyan En Büyük Alan)
TÖVD	Taban Öncelik Vektörel Doldurma

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Müşterilerin araç alımında önemli seçim kıstaslarından birisi olan bagaj hacmi araç üreticileri için önemli bir tasarım ve araç satış faktörü haline gelmiştir. Daha önce kataloglarda boyutsal olarak verilen bagaj hacimleri ilerleyen yıllarda Almanya ve Amerika menşeili araç üreticileri tarafından hacim olarak sunulmaya başlanmıştır. Bagaj hacimleri kataloglarda üreticiler tarafından geliştirilen standartlar ile hesaplanıp litre olarak belirtilmektedir. Bu standartların geliştirilmesindeki temel amaç araç kullanıcıları bagaj içerisini olabildiğince nasıl verimli kullanabilir sorusunun cevabı içindir.

Verimli bir bagaj hacminin oluşabilmesi için temel faktörlerden biride hiç kuşkusuz bagajın iç ve dış tasarımlarıdır. Tasarımı gerçekleştirilen bir bagaj sonraki süreçte hacim hesabı yapılmak üzere öncelikle CAD ortamında standartların belirtmiş olduğu kurallara göre hesaplanmaktadır. Bir sonraki adımda ise prototip olarak üretilen aracın hesaplama yöntemine göre fiziksel ve gerçek ortamda doğrulanması yapılmaktadır.

Bu üç adımda gerçekleştirilen hacim hesaplama araç üreticileri için önemli bir zaman kaybına neden olmaktadır. Üretilecek olan bir araç için hedeflenen bagaj hacmi CAD ortamda hesaplandıktan sonra istenilen hedefe ulaşmaması durumdan tekrardan tasarımın yapılmasına neden olmakta ve aynı çalışma tekrardan yapılmak zorunda kalacaktır. Yapılacak olan tasarım değişikliği sonucu araç üzerindeki farklı tasarımlarında etkilenmemesi kaçınılmaz olacaktır. Bu yöntem ile bagaj hacim hesabının yapılmasındaki en büyük negatif durum CAD ortamında manuel olarak yapılmasıdır. Çünkü işlemi yapan kişi hacmi en kısa sürede en az boşluk kalacak şekilde hatasız olarak doldurmalıdır. Bu durum, üreticileri operatörlerin yeteneğine mecbur bırakmaktadır.

Bu çalışmada operatörün deneyiminden bağımsız, dik yerleştirme kurallarına göre maksimum sayıda standart blok sığdırılacak şekilde CAD modeli üzerinde çalışan bir yerleşim algoritması geliştirilmiş ve kodlanarak örnek bagaj modellerine uygulanmıştır.

Gerçek bagaj modeli üzerinde 3-4 gün süren çalışma, CAD ortamında dakikalar içinde sonuçlanmaktadır.

2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI

Literatürde, kapalı bir hacim içine geometrilerin yerleştirilmesi ile ilgili birçok çalışma bulunmaktadır. Lojistik sektöründe tır taşımacılığında da benzer bir ihtiyaç söz konusudur. Farklı ebatlardaki taşınacak ürünlerin, tır içine en az boşluk kalacak şekilde yerleştirilerek en az sayıda tır ile taşıma yapılması istenmektedir. Böylece taşıma maliyetleri en aza indirgenip zaman ve maliyetten tasarruf edilmek istenir. Aynı şekilde araç bagaj hacminin hesaplanması içinde benzer bir durum söz konusudur. Maksimum sayıda blok yerleşimi ve minimum hacimde boşluk için optimizasyon algoritmalarını kullanan çalışmalarda mevcuttur.

2.1. Uluslararası Standartlar

Araçların bagaj hacimleri, kapladıkları sürekli geometrik hacme göre değil, uluslararası standartlarda tanımlanmış yöntemlere göre hesaplanmaktadır. Araç üreticileri de araç bilgilerini içeren kataloglara bu standartlara göre hesaplanmış değerleri yazmaktadırlar, ancak bu değerlerin standartlara göre hesaplanıp hesaplanmadığını kontrol eden bir mekanizmada henüz bulunmamaktadır.

Bagaj hacmi hesaplanması ile ilgili 3 standart yayınlanmıştır. Bunlar DIN 70020, ISO 3832 ve SAE J1100 standartlarıdır. Avrupa'da üretilen araçlar için DIN ve ISO standartları, A.B.D'de üretilen araçlar ise SAE standardına göre bagaj hacimleri hesaplanmaktadır. DIN 70020 ve ISO 3832 standartları içerik olarak aynıdır. Bu standartta yer alan metod Alman VDA kurumu tarafından geliştirilmiştir. Daha sonra DIN ve ISO standardı olarak kabul edilmiştir.

ISO ve DIN standardında kullanılan 1 litrelik dikdörtgenler prizması şeklindeki bloğun ölçüleri 200 mm, 100 mm ve 50 mm'dir (ISO 3832 2002). Bu blok, geometride küboid (cuboid) olarak isimlendirilmektedir. Araç bagajı içerisine yerleştirilmiş olan küboidler Şekil 2.1'de görülmektedir.

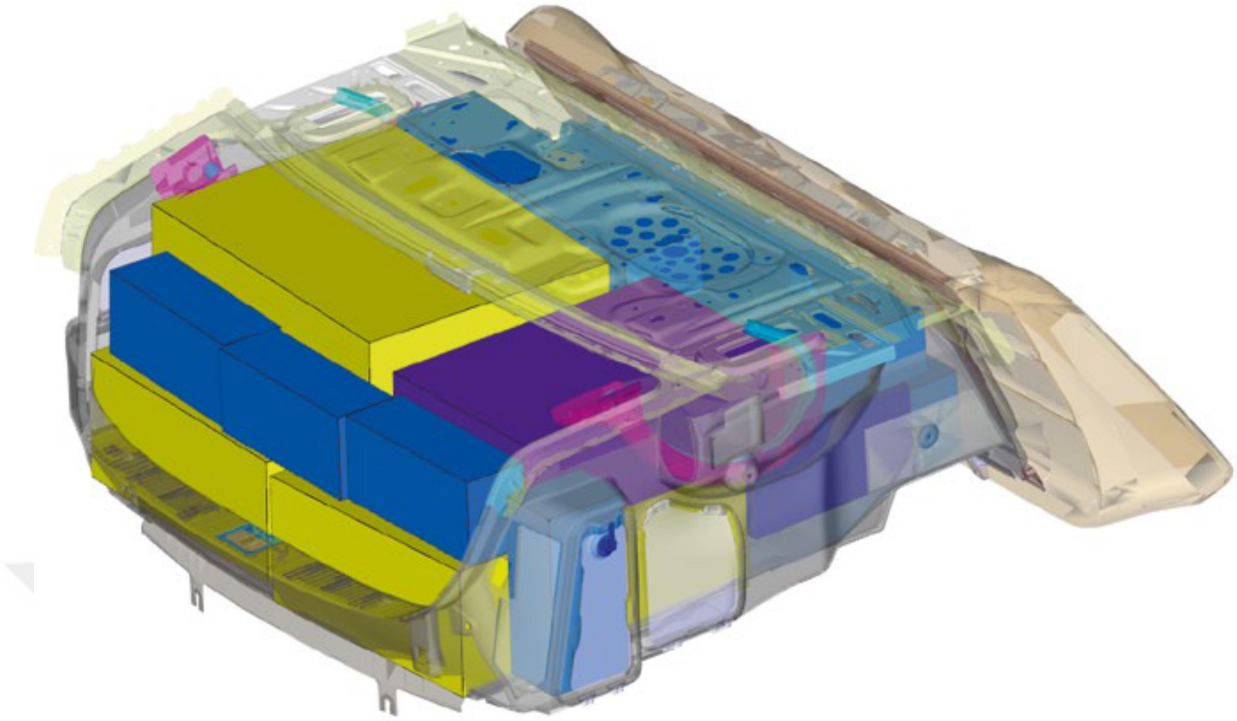
Şekil 2.1. Bagaj içerisine doldurulmuş küboidler

SAE J1100 standardında ise birden fazla blok geometrisi kullanılmaktadır. Standartta tanımlanmış sekiz farklı blok tipi vardır. Her blok tipi standardın izin verdiği sayıda kullanılarak bagaj hacim hesabı yapılmaktadır (SAE International 2011). Bu standartta yer alan blok boyutları Çizelge 2.1'de verilmiştir.

Çizelge 2.1. SAE standardı blok ölçüleri (SAE International 2011)

Blok tipi	Maksimum Adet	Uzunluk (mm)	Genişlik (mm)	Yükseklik (mm)
A	4	610	483	229
B	4	457	330	159
C	2	660	406	229
D	2	553	457	216
E	2	381	229	203
F	2	533	356	178
G	2	1143	204	204
H	20	325	152	114

SAE standardında kullanılan blok tipleri günlük yaşantımızda kullanılan bavul tip ve ölçülerine yakındır. Bu standart, kullanılan bavul boyutlarında bloklar içerdiğinden günlük kullanıma daha uygun bir hacim hesabı yapmaktadır. SAE standardına ait bagaj içerisine yerleştirilmiş bloklar Şekil 2.2'de görülmektedir.



Şekil 2.2. SAE standartlarına göre doldurulmuş bagaj örneği (Dziegielewski ve Erbes 2016)

2.2. CAD Algoritmaları

CAD algoritmaları eldeki tasarımları ve verileri tanımlamamız için gerekli bir elemandır. Mühendislik tasarımları için kullanılan CAD paket programları yazılım firmaları tarafından geliştirilmiştir. Kullanılan programlar içerisinde operatörlere kolaylık olması amacıyla birçok geliştirme yapılmıştır. Bunlardan bir tanesi de yazılım dili kullanarak bir işlem gerçekleştirmektir. Program içerisinde yer alan ve programda çalışma yaparken kullanıcıların göremediği ve program alt tabanında sürekli olarak çalışan bir yazılım mevcuttur. Bu yazılım kullanılarak sürekli rutin bir şekilde yapılmakta olan işlemlerimizi yazılım dilini kullanarak kaydedildiğinde dakikalar veya saatler süren işlemler artık tek bir komut ile yapılabilir hale gelmektedir. Günümüzde geliştirilmiş olan birçok CAD programları farklı yazılım dil seçeneklerini içerisinde barındırmaktadır.

NX CAD programı içerisinde barındırmış olduğu NX Open yazılımı ile kullanıcılara yapılacak olan işlemlerde kolaylık sağlamaktadır. NX Open, NX içerisinde program

yazmamızı sağlayan uygulama arabirim geliştiricisidir. Sürekli olarak yapılan tasarımların ve diğer işlemlerin hızlı bir şekilde birkaç komutla yapılmasını sağlar. NX Open kullanıcılara farklı işlem kabiliyetleri sunan geniş bir fonksiyona sahiptir. Bunlardan bazıları şu şekildedir:

- Parça geometrisi, montaj, çizim
- Parça bilgilerini okumak, üzerinde bir takım işlemler yapmak,
- Kullanıcıların dosya seçmesine ve veri girmesine olanak sağlayan kullanıcı ara yüzü oluşturmak gibi.

Bu işlemlerin bazı uygulamaları ise şunlardır:

- NX dışındaki diğer kaynaklardan içe veri aktarmak,
- Part bir dosyaya ait verileri okuyup dosya halinde sunmak,
- Bir takım işlemleri bir araya getirerek özel bir uygulama oluşturmak.

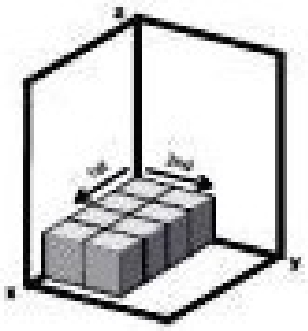
Bahsedilen bu işlemlerin ana teması sürekli hale gelen çalışmalara harcanan zamanı minimum seviyeye çekerek işlemleri otomatikleştirmektir.

Bagaj hacmi hesaplama çalışmasında CAD programın sunmuş olduğu bu avantajlar kullanılarak manuel olarak yapılan işlemlerin hızlandırılması mümkün kılınmış ve kullanıcı hatalarının önüne geçilmesi sağlanmıştır. Hacim hesaplamasında bagaj içerisine yerleştirilmiş olan küboid sayısına göre belirlendiğinden küboidlerin birbiri üzerine montajlanması ve çakışması durumu hacim hesabında önemli bir husustur. Manuel olarak yapılan çalışmalarda göz ile kontrolde hata payı her zaman var olacaktır fakat NX Open uygulaması içerisinde yer alan özellik bu hatanın önüne geçilmesine olanak sağlayacaktır.

2.3. Yerleşim Algoritması

Yerleşim algoritmaları bagaj içerisine yerleştirilecek olan nesnenin yerleşim sırasını ve düzenini belirlemek için kullanılır. Algoritmanın temel amacı bagaj hacmi verimini artırmak için yerleştirilen nesnelerin en uygun olan şekli ile yerleştirmektir. Bu amaç doğrultusunda geliştirilmiş olan uygulamalar arasında alt-sol-köşe-doldurma (bottom-left-back-fill) BLBF yöntemi kullanılmaktadır. BLBF yöntemi aşağıdaki adımları izlemektedir (Tiwari 2009):

- Birinci nesneyi sol alt arka köşeye yerleştir.
- Sıradaki eklenen parçayı mümkün olan en alt sol arka köşeye yerleştir.
- Eklenen her nesne için bu yolu izle.
- Böyle bir konum bulunmazsa nesneyi ekleme.



Şekil 2.3. Katman oluşturma yaklaşımı (Domingo ve ark. 2013)

BLBF sezgisel yöntemi ile hareket sol-alt-arka-köşe noktadan başlayarak sağ-üst-ön-köşe noktasına kadar parçalar yerleştirilerek uygulanır (Şekil 2.3). Tüm bagaj hacmi doldurulduktan sonra işlem sonlandırılır.

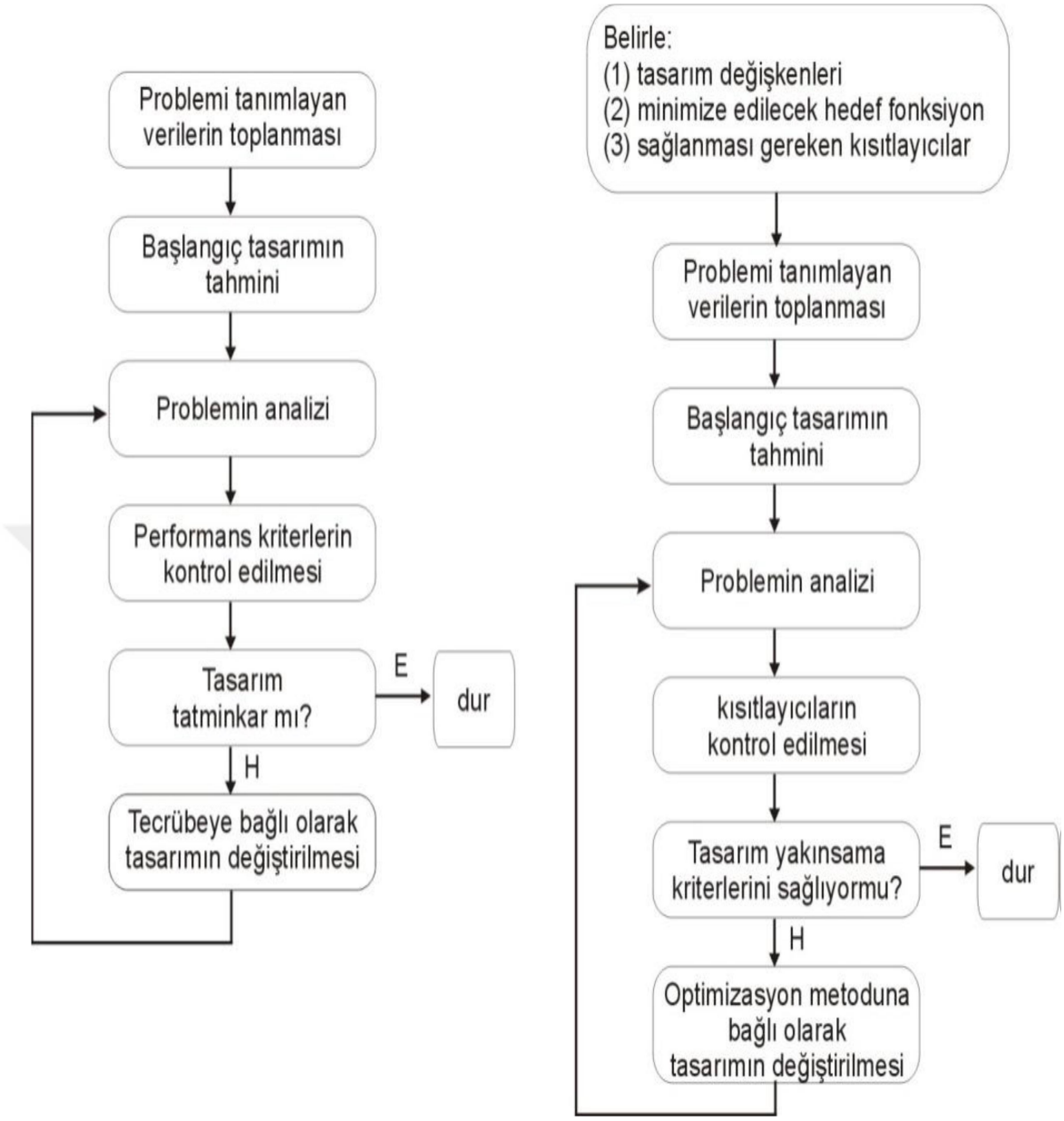
2.4. Optimizasyon Algoritması

Optimizasyon, en iyileme anlamına gelmektedir. Bir problem için, verilen şartlar altında tüm çözümler arasından en iyi çözümü elde etme işidir. Belirli sınırlamaları sağlayacak şekilde, bilinmeyen parametre değerlerinin bulunmasını içeren herhangi bir problem, optimizasyon problemi olarak adlandırılabilmektedir (Murty 2003).

Kalınlı (2012) optimizasyon konusunda ki görüşlerini "optimizasyon metotları iki düzeyde tanımlanır; problem düzeyinde ve problem üstü düzeyde. Problem düzeyinde optimizasyon işleminde, bir problem kümesi içerisindeki her problem tek tek diğerlerinden bağımsız olarak çözülür. Problem üstü düzeyde çeşitli problemlerden oluşan bir küme bir bütün olarak göz önüne alınarak optimum çözüm aranır. Problem

üstü düzeyde, optimizasyon metotları, problem düzeyindeki optimizasyon metotlarını yönlendirmek amacıyla da kullanılır" şeklinde ifade etmektedir.

Kalınlı (2012) optimizasyon problemleri hakkında "muhtemel farklı çözümleri var olan ve çözümlerin kalitesi hakkında açıkça fikir sahibi olunabilecek basit bir yapıda olabilir. Böyle bir problemin farklı aday çözümleri var olduğunda, bunlar anlamlı bir şekilde ayrıştırılabilir ve karşılaştırılabilir. Bununla beraber, bir çok optimizasyon problemi esas itibariyle zordur. Endüstri veya bilimle ilgili problemler ve bir çok gerçek problemde tipik senaryo problemin zor olmasıdır. Örneğin bir lojistik firması maksimum taşıma kapasitesi sağlayarak minimum harcamayı nasıl gerçekleştirebilir?" şeklinde bir ayrıştırma tanımlaması yapmıştır. Bu ve buna benzer problemler içerisinde birçok değişken ve kısıtlama bulunmaktadır. Manuel olarak problem çözme işlemi operatör tecrübesine ve becerisine bağlı olduğundan elde edilen sonucun optimuma yakın olması gerçekçi bir çözüm oluşturmayabilir (Şekil 2.4).



Şekil 2.4. Klasik yaklaşım ile optimizasyon yaklaşım karşılaştırması (Kaymaz 2018)

Kalınlı (2012) "sezgisel stratejiler, genellikle problem üstü düzeyde kullanılır. Literatürde heuristic olarak tanımlanır. Sezgisel terimi; akıllı tahminlere dayalı buluş yöntemi veya yeni çözümlerin keşfine götüren bulgulara dayalı arama yöntemi olarak tanımlanır. Sezgisel terimi ile tanımlanan algoritmaların ortak özelliği; araştırma uzayı çok büyük olan problemlerde çözümün aranmasını sınırlayan bir kural, strateji, hile, sadeleştirme ve benzeri etmenler kullanmalarıdır. Bu etmenlerden genellikle araştırmayı yönlendiren bir bulgu kastedilir. Bulgucu stratejiler arama zamanını kısaltır, optimum

çözüme yakın çözümler sunar fakat optimum çözümü garanti edemez" ifadesi ile optimizasyon çözümlerinde kullanılan sezgisel yönteme dair görüşlerini belirtmiştir.

Karaboğa (2004) sezgisel algoritmaların önemini "sezgisel algoritmalar, herhangi bir amacı gerçekleştirmek veya hedefe varmak için doğal fenomenlerden esinlenen algoritmalardır. Bu algoritmaların, çözüm uzayında optimum çözüme yakınsaması ispat edilememektedir. Yani sezgisel algoritmalar yakınsama özelliğine sahip olmaktadır, ama kesin çözümü garanti edememektedir ve bu kesin çözümün yakınlarında bir çözüm garanti edebilmektedir. Anlaşılırlık yönünden sezgisel algoritmaların karar verici açısından çok daha basit olabilmesinden, optimizasyon problemlerinin kesin çözümü bulma işleminin tanımlanamadığı bir yapıya sahip olmasından ve öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanabilirliğinden sezgisel algoritmalara ihtiyaç duyulmaktadır" ifadesi ile ortaya koymaktadır.

Yaklaşık son 30 yıldan bu yana basit sezgisel algoritmaları daha yüksek bir seviyede birleştirerek arama uzayını daha verimli ve etkin bir şekilde keşfetmeyi amaçlayan yöntemler ortaya çıkmıştır. Bu yöntemler günümüzde metasezgisel algoritmalar olarak da bilinmektedir. Metasezgisel, optimal sonuca yakın çözümleri verimli bir şekilde bulabilmek için arama uzayında global ve yerel arama sürecini sezgisel yöntemlerin zeki davranışlarla ve öğrenme stratejileriyle yönlendirdiği yinelemeli (iteratif) üretim işlemi olarak tanımlanır (Osman ve Laporte, 1996). Birçok metasezgisel algoritma doğadan esinlenerek ortaya çıkarılmıştır. Metasezgisel algoritmalar farklı türlerdeki problemlere çözüm sunma kabiliyetine sahiptirler. Yapılan çalışmalarda geliştirilmiş olan bazı metasezgisel algoritmalar mevcuttur. Genetik Algoritma (GA), Diferansiyel Gelişim Algoritması (DGA) ve Tavlama Benzetimi (Simulated Annealing – SA) bunlardan bazılarıdır.

2.4.1. Genetik algoritma

Genetik algortima (GA), metasezgisel optimizasyon algoritmalar sınıfında yer alan ve problem çözümlerinde başarıyla kullanılan bir optimizasyon aracıdır. GA uygun genetik varyasyonlar sağlandığı sürece her türlü çözüm işleminde kullanılabileceğinden çok

fonksiyonlu optimizasyon problemleri için optimizasyon algoritmaları tasarımı sunar. Makine öğrenmesi konusunda çalışan Michigan Üniversitesinde psikoloji ve bilgisayar uzmanı olan John Holland (1975) bu konuda ilk çalışmaları yapan kişidir. GA, doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir (Özkan 2003). Bunun için:

- İyi'nin ne olduğunu belirleyen bir uygunluk (fitness) fonksiyonu
- Yeni çözümler üretmek için yeniden kopyalama (recombination)
- Değiştirme (mutation)

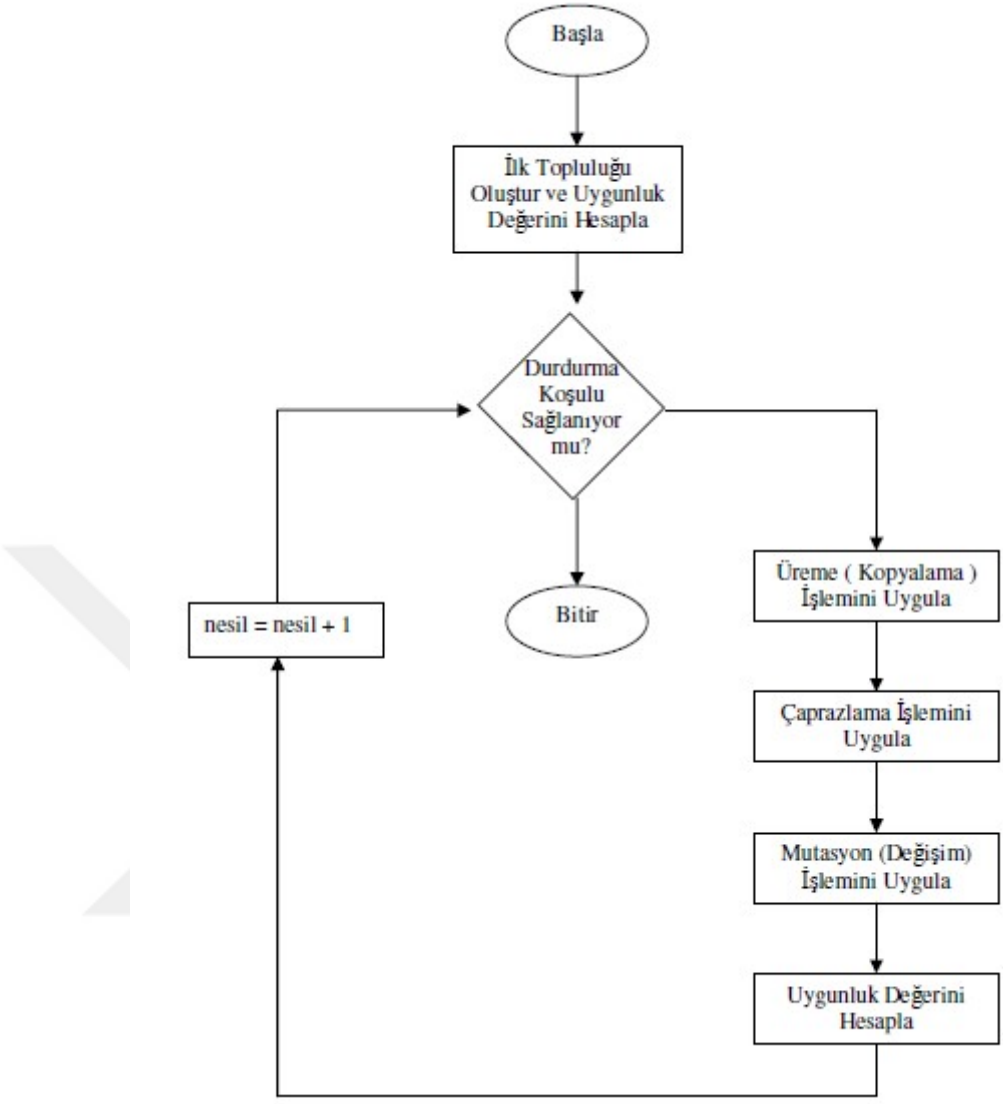
gibi operatörleri kullanır.

GA parametreleri, biyolojideki genleri temsil ederken, parametrelerin toplu kümesi de kromozomu oluşturmaktadır. GA'ların her bir ferdi kromozomlar (bireyler) şeklinde temsil edilen popülasyonlardan oluşur. Popülasyonun uygunluğu, belirli kurallar dâhilinde maksimize veya minimize edilir. Her yeni nesil, rasgele bilgi değişimi ile oluşturulan diziler içinde hayatta kalanların birleştirilmesi ile elde edilmektedir (Çalışır 2015).

Mutluer (2007) GA'ların avantajlarını şu şekilde belirtmiştir;

- Sürekli ve ayrık parametreleri optimize etmesi
- Türevsel bilgiler gerektirmemesi
- Amaç fonksiyonunu geniş bir spektrumda araştırması
- Çok sayıda parametrelerle çalışma imkânı olması
- Paralel PC'ler kullanılarak çalıştırılabilmesi
- Karmaşık amaç fonksiyonu parametrelerini, lokal minimum veya maksimumlara takılmadan optimize edebilmesi,
- Sadece tek çözüm değil, birden fazla parametrelerin optimum çözümlerini elde edebilmesi olarak sıralanabilir.

Şekil 2.5'te genetik algoritma akış şeması belirtilerek çalışma prensibi gösterilmiştir.



Şekil 2.5. Genetik algoritma akış şeması (Şeker 2008)

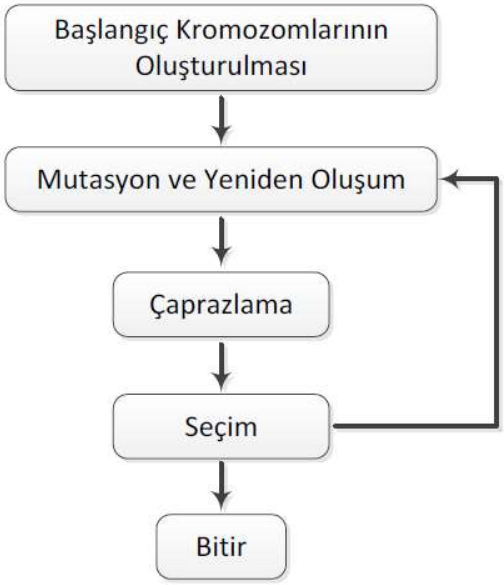
2.4.2. Diferansiyel gelişim algoritması

DGA, Price ve Storn (1997) tarafından geliştirilmiş, özellikle sürekli verilerin söz konusu olduğu problemlerde etkin sonuçlar verebilen, işleyiş ve operatörleri itibariyle genetik algoritmaya dayanan popülasyon temelli sezgisel optimizasyon tekniğidir. Tamamen düzenlenmiş uzayda tanımlı ve gerçek değerli tasarım parametrelerini içeren fonksiyonları küresel olarak optimize etmek amacıyla kullanılan bir direkt araştırma algoritmasıdır (Karaboğa 2004). DGA özellikle numerik optimizasyonlar üzerinde etkili olan bir gelişim algoritmasıdır. Bu algoritma yeni, ama sadece basit olmayan aynı

zamanda oldukça da etkili olan bir mutasyon işlemi uygulamaktadır. Daha önce tanımlanmış, olasılık dağılım fonksiyonuna dayalı olarak çalışan genetik algoritma gibi gelişim tabanlı algoritmaların tersine diferansiyel gelişim algoritması rastgele olarak seçilmiş amaç vektör çiftlerinin farklarına dayalı bir mutasyon işlemi kullanır (Karaboğa 2004).

Diferansiyel gelişim algoritmasında kullanılan basit mutasyon işlemi, algortimanın performansını geliştirmekte ve onu daha robust yapmaktadır. Bu özelliğinin yanı sıra diğer özellikleri için de şunlar söylenebilir: Hızlı, basit, kolayca kullanılabilir ve değiştirilebilir, etkili küresel optimizasyon kabiliyetli, doğal olarak paralel, kayan-nokta formatına bağlı hassasiyet sınırlamalı, matris çarpımları ve sıralama işlemleri olmadığı için hesaplama maliyeti açısından avantajlı, daha önceden tanımlanmış herhangi bir olasılık dağılımlı mutasyon kullanmamakta, tamsayı, ayrı ve karışık parametre optimizasyonuna kolaylıkla uyarlanabilir, amaç fonksiyonunun veya sınırlama fonksiyonlarının türevine gerek duymaz, düz yüzeylerde çalışabilir, gürültülü ve zamana bağlı amaç fonksiyonları için kullanılabilir, tek bir koşmada alternatif çözümler üretebilir ve özellikle doğrusal olmayan sınırlamalı optimizasyon problemlerinde etkilidir (Karaboğa 2004).

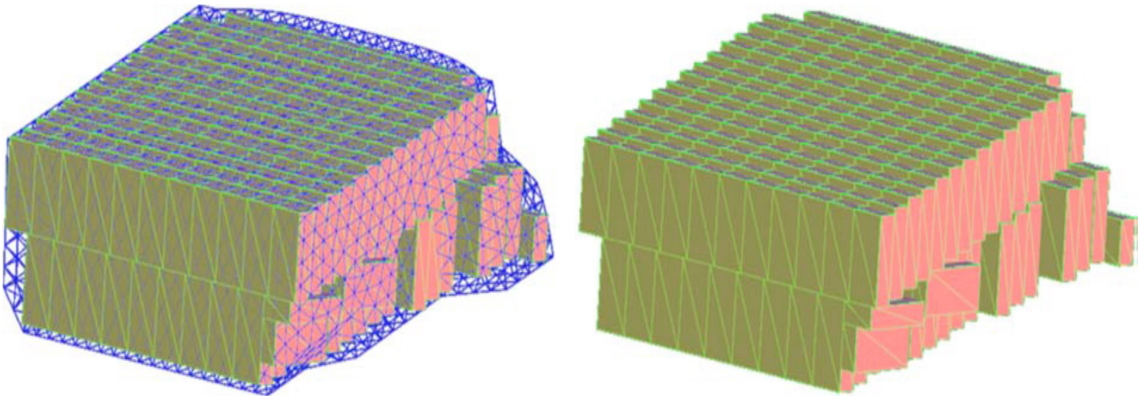
DG'nin önemli parametreleri: NP (Number of Population-popülasyon büyüklüğü), CR (Crossover Rate-çaprazlama oranı), F (ScalingFactor-ölçekleme faktörü) olarak sayılabilir. DG algoritmasında oluşturulan başlangıç toplumu mutasyon, çaprazlama ve seçim operatörleri kullanılarak en iyi değerin elde edilmesi amacıyla iterasyonlar boyunca iyileştirilmektedir (Liu ve ark. 2010). Toplum büyüklüğünü temsil eden NP parametresi bir problem için dikkate alınan çözüm vektörlerinin sayısını, algoritma içinde kullanılan bir diğer kontrol parametresi F ile toplum içinden rastgele seçilen ve birbirinden farklı 3 adet çözüm vektöründen yeni bir vektör üretilmesi amacıyla, DG algoritmasında kullanılan son kontrol parametresi çaprazlama oranında (CR) ise bu mutasyon sonucu elde edilen vektörün dikkate alınma oranı olarak kullanılmaktadır (Başkan ve Ceylan 2014). Algoritmaya ait akış şeması Şekil 2.6'da gösterilmiştir.



Şekil 2.6. DGA'nin akış diyagramı (Özyön ve ark. 2011)

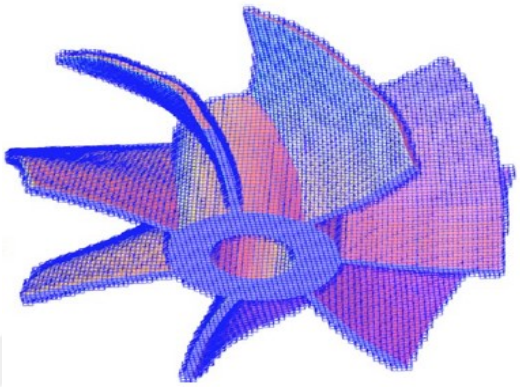
2.5. Literatür Taraması

Tiwari ve ark. (2010), ISO 3832 ve SAE J1100 standartları çerçevesinde örnek bir bagaj modelini ele alarak her iki standart içinde çalışma yapmışlardır. Yapmış oldukları çalışmada optimizasyon algoritması, yerleşim algoritması ve CAD algoritmalarını birbiri ile bağlantılı bir şekilde çalıştıracak bir yöntem ile örnek bagaj hacmi için en yüksek verim sonucunu çözümlemişlerdir (Şekil 2.7).



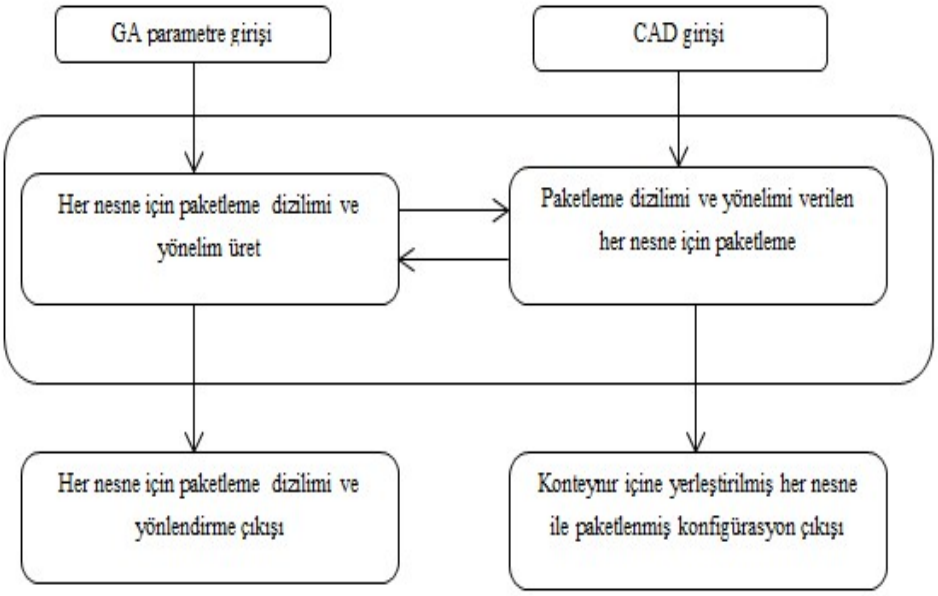
Şekil 2.7. ISO standardında küboid yerleşimleri (Tiwari ve ark. 2010)

SAE standardı için yapılan çalışmada ek olarak CAD algoritması kullanmışlardır. Tiwari ve ark. (2010) CAD algoritmasında objeleri literatürde voxel olarak adlandırılan üç köşeli katı elemanlara ayırmışlardır (Şekil 2.8). Bagaj içerisine konulan objelerin birbiri ile girişim yapıp yapmadığını hızlı bir şekilde tespit etmek için kullanılmış bir yöntemdir.



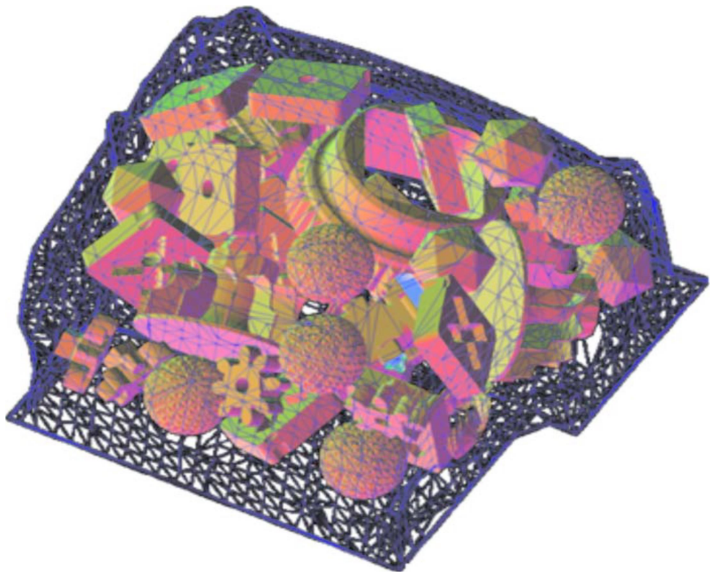
Şekil 2.8. Basit voxelleştirilmiş nesne (Tiwari ve ark. 2010)

Tiwari ve ark. (2010) bu çalışmada optimizasyon çözümü için genetik algoritma, yerleşim algoritması için ise BLBF yöntemi ve yerleşim sezgiselliğini kullanmıştır. Şekil 2.9'da çözüm stratejilerine ait akış şeması gösterilmiştir.



Şekil 2.9. Çözüm stratejisi kavramsal akışı (Tiwari ve ark. 2010)

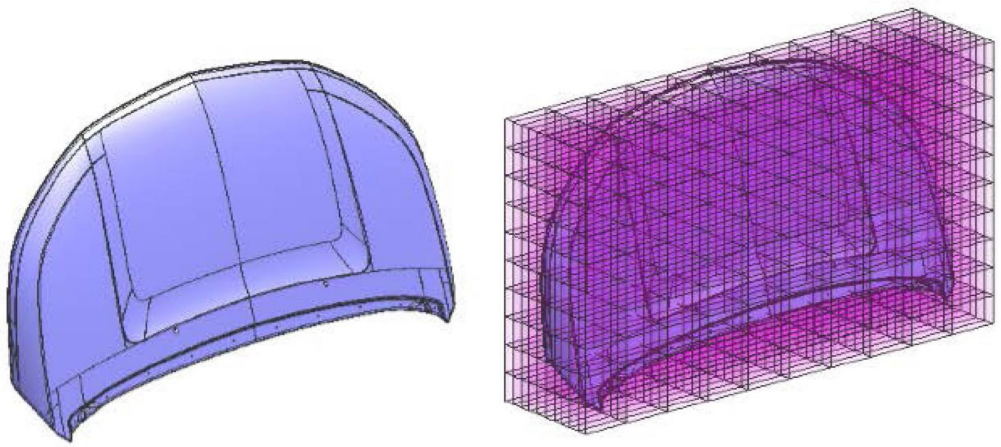
Her iki standart içinde gerçekleştirilen çalışmada sağlıklı bir sonuç elde edebilmek ve karşılaştırma yapabilmek için aynı bagaj modeli kullanılarak sonuçlar gözlemlenmiştir. Sonuçlar önerilen paketleme algoritmasının sağlıklı çalıştığını ortaya koymuştur (Şekil 2.10).



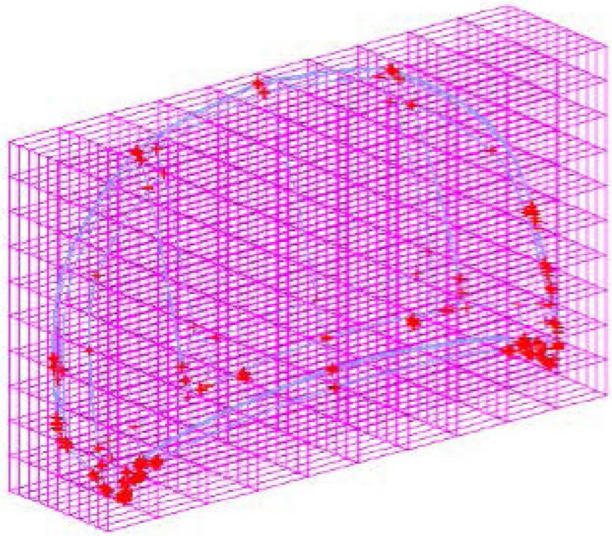
Şekil 2.10. Tamamen doldurulmuş bagaj (Tiwari ve ark. 2010)

Domingo ve ark. (2013), konteynır yükleme problemini ele alarak diferansiyel gelişim algoritması ve yerleşim algoritmasını kullanarak konteynır içerisine üç farklı tip blokları yerleştirmişlerdir.

Joung ve Noh (2014) araç yükleme problemi için gruplama algoritması tabanlı bir çalışma gerçekleştirmişlerdir. Yapmış oldukları çalışmada bir birleriyle etkileşimli dört adet algoritma kullanmışlardır. Bunlar gruplama algoritması, sıralama algoritması, yönelim algoritması ve yükleme algoritmasıdır. Kurgulamış oldukları gruplama algoritmasında sınırlayıcı kutu tekniği adı verdikleri bir yöntem geliştirmişlerdir. Sınırlayıcı kutu tekniği yöntemine göre bagaj içerisine konulacak olan nesne etrafına sınırlayıcı bir kutu yerleştirilir ve kutu 1000 küçük hücreye bölünür (Şekil 2.11). Nesnenin kenarlarına denk gelen hücrelerde noktalar oluşturulur (Şekil 2.12). Daha sonra nesnenin kenarı boyunca oluşturulan noktaların dağılımı karşılaştırılarak parçanın hangi gruba girmesi gerektiğine karar verir. Gruplaması yapılan nesnelerin sınırlayıcı kutu hacimleri bagaj içerisine yükleme algoritması kullanılarak yerleştirilir ve böylelikle hacim hesaplaması yapılmış olur.



Şekil 2.11. Örnek parça ve hücre dağılımı (Joung ve Noh 2014)



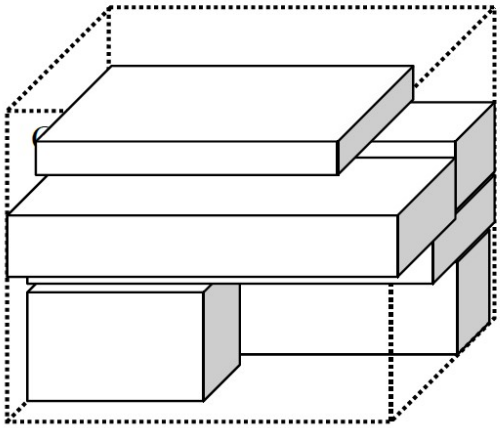
Şekil 2.12. Parça kenarlarında oluşturulan noktalar (Joung ve Noh 2014)

Shellshear ve ark. (2012) yaptıkları çalışmada tersine çevrilmiş doldurma yöntemi isimli yeni bir yöntem ile yükleme problemine çözüm aramışlardır. Hedef olarak herhangi bir bloğun herhangi bir açı ve yönelim pozisyonu şeklinde bagaj içerisine yerleşimini seçmişlerdir. Shellshear ve ark. (2015) bagaj yükleme çalışmalarında genetik algoritma ve yerel optimizasyonu birbiri ile etkileşimli şekilde programlayarak blokları bagaj içerisine yerleştirmişlerdir. Optimizasyon çalışmaları her blok için 24 farklı yönelimden birini seçerek uygun olan yönelimi genetik algoritmanın seçmesiyle yükleme yapılmış ve en verimli sonuç bulunarak bagaj hacmi belirlenmiştir.

Dereli ve Daş (2010) konteynır yükleme problemleri için karınca kolonisi optimizasyonu yaklaşımını temel alan iki yeni algoritma önermiştir. Parametreleri faktöriyel tasarım ile belirlenen bu algoritmaların performansları literatürde verilen standart problemler için test edilmiş ve sonuçlar literatürdeki diğer çalışmalar ile mukayese edilerek irdelenmiştir.

Gürbüz ve ark. (2009) konteynır yükleme problemini ele alarak boş bir konteynır hacmini efektif verimi yüksek olacak şekilde doldurma çalışması gerçekleştirmişlerdir. Yapılan çalışmada konteynır içerisine tek tip veya farklı boyutlardaki kutular olacak şekilde incelemişlerdir (Şekil 2.13). Bu çalışmada probleme özgü largest area first fit

(LAFF) adını verdikleri bir algoritma geliştirmişlerdir. Önerilen algoritmada öncelik yüksekliği en aza indirgeyerek en geniş alan kullanımı göz önüne alınmıştır.



Şekil 2.13. Algoritmanın muhtemel bir çözümü (Gürbüz ve ark. 2009)

Algoritmada öncelikle kullanılan konteynırın genişliği ve derinliği belirlenir. Verilen kutuların en uzun iki kenarı bulunarak konteynırın genişliği ve derinliği belirlenmiş olur. Daha sonra konteynır içerisine kutular yerleştirilir. Konteynır içerisine kutuların yerleştirilmesi için iki farklı yerleşim metodu eş olarak çalıştırılır. Yerleştirilecek olan farklı boyutlardaki kutular program tarafından seçilir ve konteynır doldurulur. Gürbüz ve ark. (2009) bu çalışma için 20 farklı boyutlardan oluşan toplamda 1000 adet kutu kullanmıştır. Farklı boyutların veya tek tip boyutlu kutular kullanılarak elde edilen sonuçlar Çizelge 2.2'de belirtilmiştir.

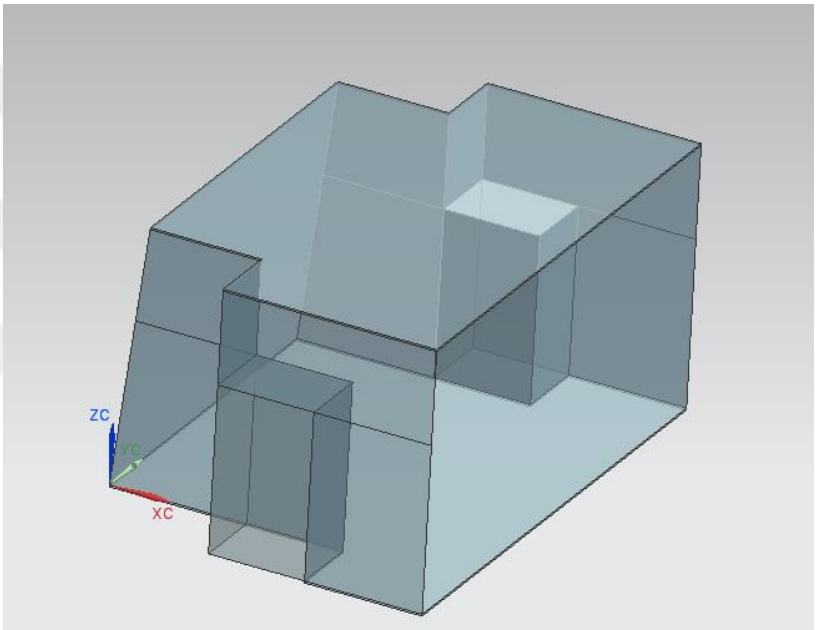
Çizelge 2.2. Örnek hacim hesaplama sonuçları (Gürbüz ve ark. 2009)

Farklı kutu sayısı	Toplam kutu sayısı	Yerleştirilen kutuların sayısı	Konteynır hacmi	Boş kalan hacim (%)
1	10	8400	8400	0
1	20	12480	12480	0
2	5	2547	2610	2,41
2	10	6252	6480	3,52
2	15	25554	25920	1,41
2	20	49032	49320	0,58
5	5	3910	5100	23,3
5	10	11359	13680	16,97
5	20	22596	25840	12,55
10	10	13419	19200	30,11
10	20	21694	27740	21,80
10	30	12854	16800	23,49

3. MATERYAL VE YÖNTEM

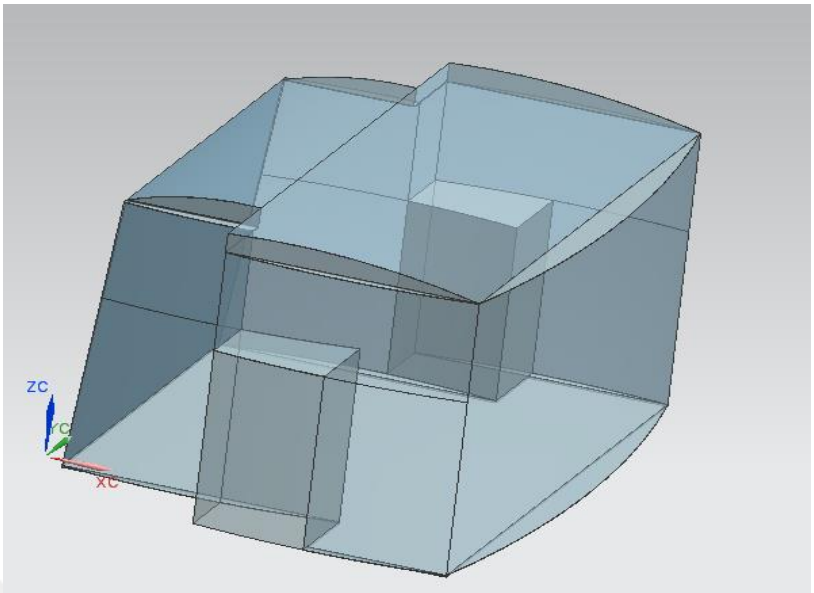
3.1. Materyal

Bagaj modelleri için iki farklı araç örnek alınmıştır. Araçlardan birisi sedan araç modeli bir diğeri ise ticari araç modelidir. Araçların gerçek bagaj tasarımları elde edilemediğinden taslak olarak iç boyutları ölçülerek ortaya konulan tasarım kullanılmıştır. Oluşturulan tasarımlarda düzlemsel yüzeyler kullanılmıştır. Sedan araç modelinde ayrıca yüzeylere form verilerek hesaplama yapılmıştır.



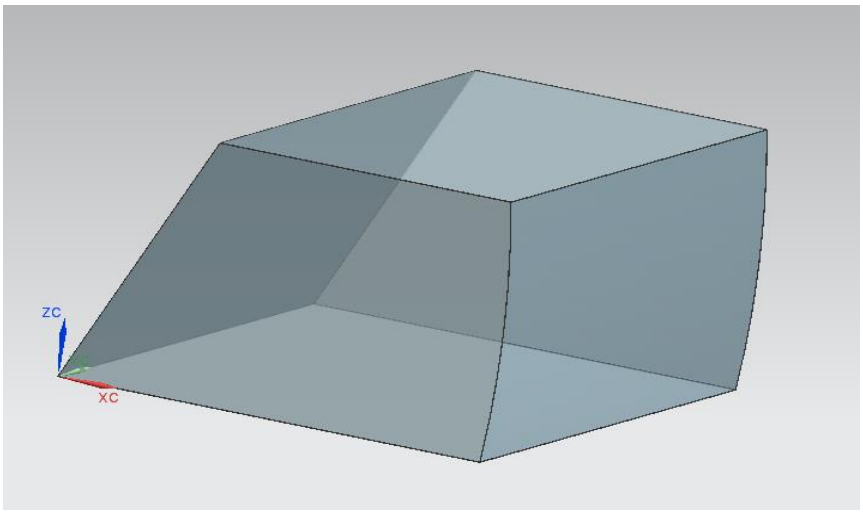
Şekil 3.1. Düzlemsel yüzeyli sedan araç örnek bagaj modeli

Şekil 3.1'de sedan araç için oluşturulan bagaj hacminin yüzey formu görülmektedir. Model yüzeylerden oluşmakta ve kapalı formdadır. Tekerlek davlumbaz çıkıntılarıda bagaj içinde dikkate alınmıştır. Arka tarafta görülen eğik yüzey koltuk tarafını göstermektedir.



Şekil 3.2. Yüzeylere form verilmiş sedan araç örnek bagaj modeli

Sedan araç için düzlemsel yüzeylerden gerçek bagaj modeline uyan bazı yüzeyler eğrisel formda modellenerek algoritmanın eğrisel yüzeylerde de çalışması test edilmiştir. Eğrisel yüzeylere ait resim Şekil 3.2'de görülmektedir.



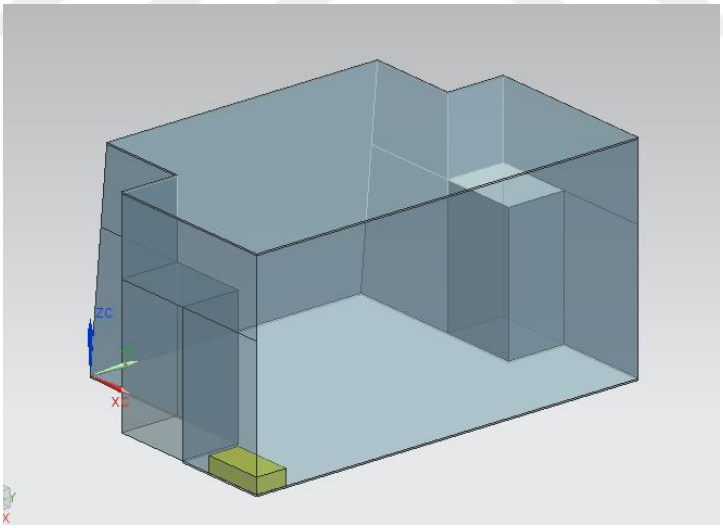
Şekil 3.3. Ticari araç örnek bagaj modeli

Geliştirilen algoritmanın test edileceği 3. bagaj modeli Şekil 3.3'te verilmiştir. Bu model ölçüleri gerçek modelden alınan ticari bir aracın basitleştirilmiş bagaj modelidir. Bu modelde düzlem yüzeyler ve eğrisel yüzeyler bulunmaktadır.

3.2. Taban Vektörel Öncelik Doldurma Yöntemi

Dış yüzeyleri verilen bir bagaj içine ISO ve DIN standardında yer alan blok geometrisinin seçilen yerleşim algoritmasına göre otomatik olarak dolduran ve CAD yazılımı içinde çalışan bir algoritma geliştirilmiştir. Doldurma sırasında blokların bagaj yüzeyi ile kesişmemesi ve blokların birbirleri içine girişim yapmaması kısıtları dikkate alınmıştır. Literatürde blokların bagaj içine yerleştirilmeye başlanacağı yer ve sonraki blokların yerleştirme sırası ve yönleri için farklı yerleştirme algoritmaları geliştirilmiştir. Bunlar arasında en yaygın olanı alt-sol-arka-doldurma algoritmasıdır. Alt-sol-arka-doldurma algoritması, bloğun bagajın sol alt arka köşesinden başlatarak sonraki eklenen her bloğun aynı düzlemde sıra ile doldurulur. İlk katman doldurulunca bir üst katmana geçilir. Tüm bloklar sıra ile yerleştirilerek dolum işlemi tamamlanır.

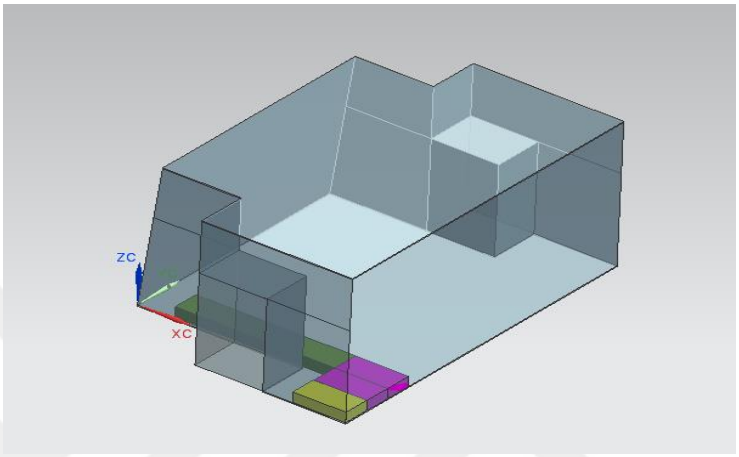
Bu çalışmada bagaj doldurma algoritması olarak yapılan çalışmalara benzer olarak sağ-alt-ön yerleştirme yöntemi seçilmiştir. Şekil 3.4'te örnek bir bagaj boşluğu içerisine sağ-alt-ön yöntemine göre eklenen ilk blok görülmektedir.



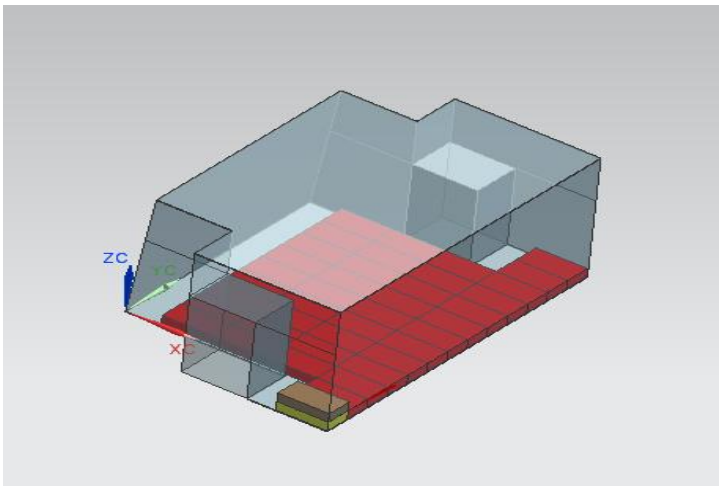
Şekil 3.4. Örnek bagaj modeli içerisine ilk eklenen blok

Bagajın içinin tamamen doldurulması için taban vektörel öncelik doldurma (TVÖD) adını verdiğimiz bir yaklaşım ile çalışma yapılmıştır. Bu algoritma blokların öncelikle bagaj tabanında X vektörü yönünde eklenmesi, ilk katman dolunca devamında Y

vektörü yönünde bir blok eklenerek yeniden X vektörü yönünde eklenmesidir. Her katman dolunca bir üst katmana geçilir. Şekil 3.5' te X ve Y yönünde doldurulmakta olan bloklar gösterilmiştir. Bagaj tabanın doldurulduktan sonra Z vektörü yönünde yeni bir blok eklenerek tekrar X ve sonra Y vektörleri yönünde bloklar eklenmeye devam eder. Şekil 3.6'da ilk katmanın tamamen dolduğu durum gösterilmiştir.



Şekil 3.5. X ve Y vektörü yönünde doldurulmakta olan bagaj

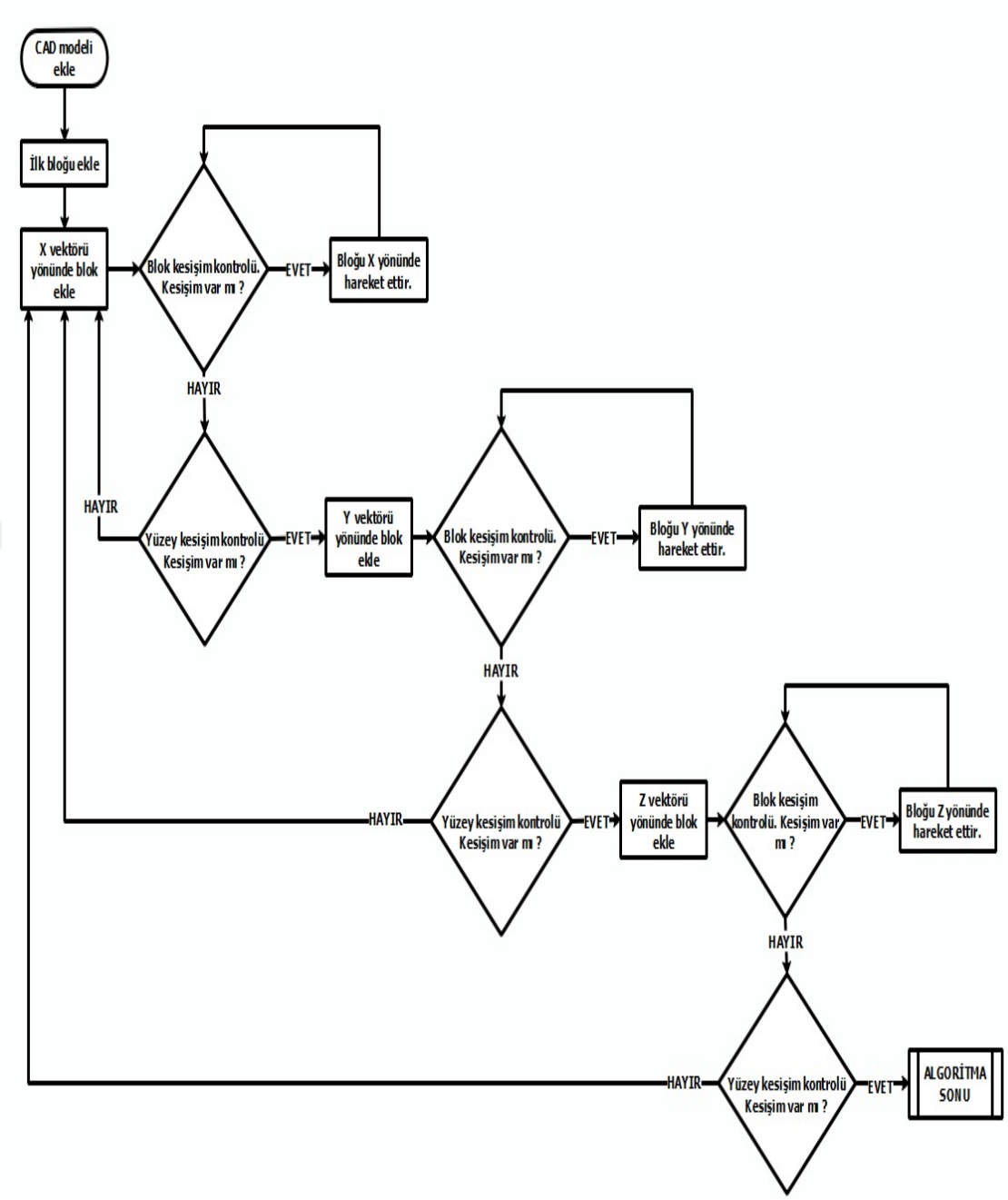


Şekil 3.6. İlk katmanı dolmuş ve bir üst katmana geçilmiş aşama

Bu algoritma referans alınarak verilen bagaj yüzey geometrisi için NX CAD yazılımı içinde çalışan bir yazılım geliştirilmiştir. Birçok CAD yazılımı, kullanıcı arayüzü ile interaktif çalışma imkânı vermesinin yanında, programlama ile kullanıcıların kendi

komutlarını geliştirebilme ve rutin yapılan işlemleri programlayarak çalışma verimini artırıcı yönde özellikler sunmaktadırlar. Bu yazılımlardan NX yazılımı da kullanıcılara NX Open adında API denilen arayüz içinde çalışarak kendi komutlarını geliştirebilme imkânı vermektedir. Böylece kullanıcılar rutin yapılan ve uzun zaman alan CAD işlemlerini programlamak sureti ile kısa zamanda gerçekleştirebilmektedirler. Bu çalışmada, NX ortamında NX Open yapısı kullanılarak Visual Basic dilinde program yazılmış ve örnek bagaj modeli üzerinde ISO ve DIN standartlarında bagaj hacim hesaplaması yapılmıştır.

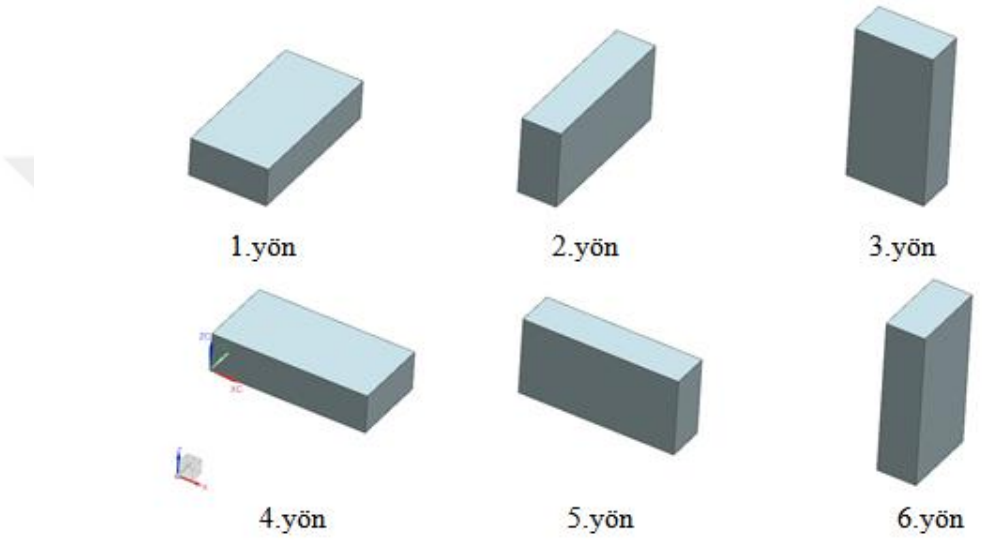
Doldurma sırasında eklenen blok başka bir blok ile kesişiyorsa eklendiği vektör yönünde kesişmeyinceye dek hareket ettirilerek blokların birbirleri ile kesişmesi engellenmiştir. Benzer şekilde blok geometrisinin bagaj yüzeyi ile kesişim yapması da engellenmiştir. Bu çalışmada geliştirilmiş olan algoritmanın akış diyagramı Şekil 3.7'de verilmiştir.



Şekil 3.7. Bagaj hacmi hesaplaması için geliştirilen algoritmanın akış diyagramı

4. BULGULAR VE TARTIŞMA

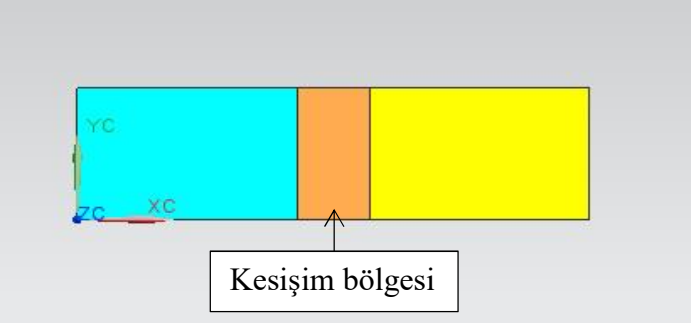
ISO ve DIN standardına göre basitleştirilmiş örnek bagajlar üzerinde hacim hesaplaması yapılmıştır. Şekil 4'te verilen algoritmaya göre hesaplama için iç yüzeyleri gerçek ölçülere yakın örnek sedan ve ticari bagaj modeli ele alınmıştır. Blokların bagaj içine yerleştirilmesi sırasında konumu 6 farklı şekilde olabilir. Bu konumlar Şekil 4.1'de görülebilir.



Şekil 4.1. Blokların farklı yönelimleri

Manuel olarak yapılan doldurma işlemlerinde blokların birbiri ile girişim yapması veya birbiri üstüne binmesi hacim hesabının yanlış yapılmasına neden olmaktadır. Doldurma işleminden sonra her blok için yapılan girişim kontrolü operatörlerin uzun zamanını almaktadır. Girişim kontrolünde yapılması muhtemel olan yanlışlıklar veya ihmaller test işlemlerinin tekrarlanmasını gerektirmektedir. Bu çalışmada ise yanlış hesaplamanın ve blokların birbiri ile girişimine izin vermemek için CAD programın sunmuş olduğu girişim kontrolü özelliği sayesinde bahsedilen durumun önüne geçilmiştir. Şekil 4.2'de gösterilmiş olan blok girişiminde eklenmiş olan blok öncelikle kesişim kontrolüne tabi tutulup kesişim olması halinde eklendiği vektör yönünde 1 mm kaydırılarak tekrardan kesişim kontrolü yapılmaktadır. Kontrol sonrasında eğer kesişim yok ise bir diğer blok

eklenmektedir. Bu sayede bagaj hacmi doğru bir şekilde hesaplanır ve operatörler için uzun zaman alan ihmallerin önüne geçilmiş olunur.

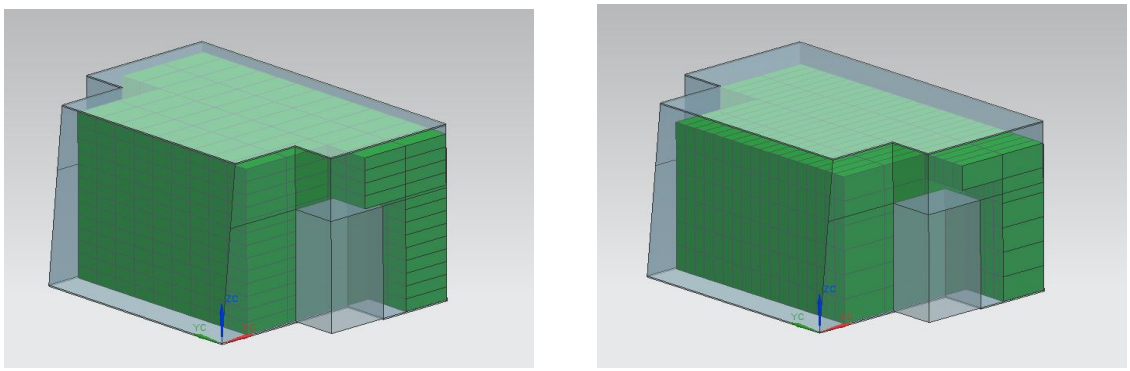


Şekil 4.2. Girişim yapan bloklar

Bir bloğun bagaj içerisine 6 farklı yönelime göre yerleştirilmesi durumları için örnek bagaj modelleri üzerinde uygulama yapılmıştır. Blokların yerleştirilmesinde ana hedef, dolum işleminden sonra bagaj içinde en az boşluk kalacak şekilde blokların yerleştirilmesidir.

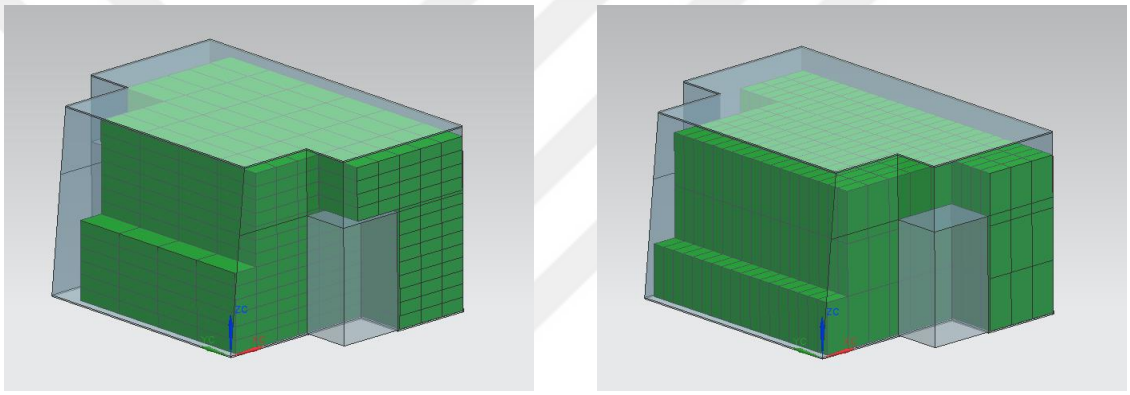
4.1. Düzlemsel Yüzeyli Sedan Araç Bagaj Hacmi

Bu bölümde, geliştirilen algoritma düzlemsel sedan araç bagajı için çalıştırılarak farklı yönlerde (Şekil 4.1.) doldurma durumları için toplam küboid sayıları elde edilmiştir. 6 farklı yön için elde edilen doldurulmuş bagaj hacim görüntüleri Şekil 4.3, Şekil 4.4. ve Şekil 4.5'te verilmiştir.



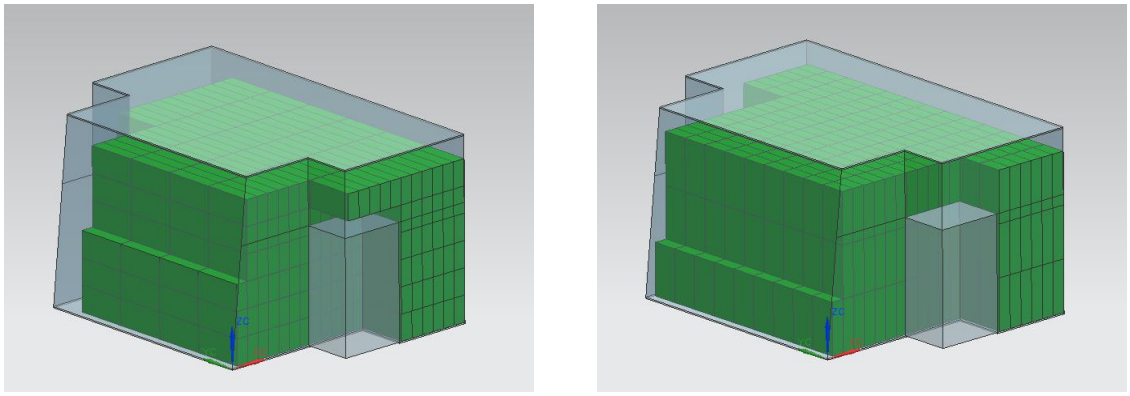
a) 1’inci yönde doldurma b) 2’nci yönde doldurma

Şekil 4.3. Küboidlerin 1’inci ve 2’nci yönlere göre düzlemsel sedan araç bagajı içine yerleştirilmesi



a) 3’üncü yönde doldurma b) 4’üncü yönde doldurma

Şekil 4.4. Küboidlerin 3’üncü ve 4’üncü yönlere göre düzlemsel sedan araç bagajı içine yerleştirilmesi



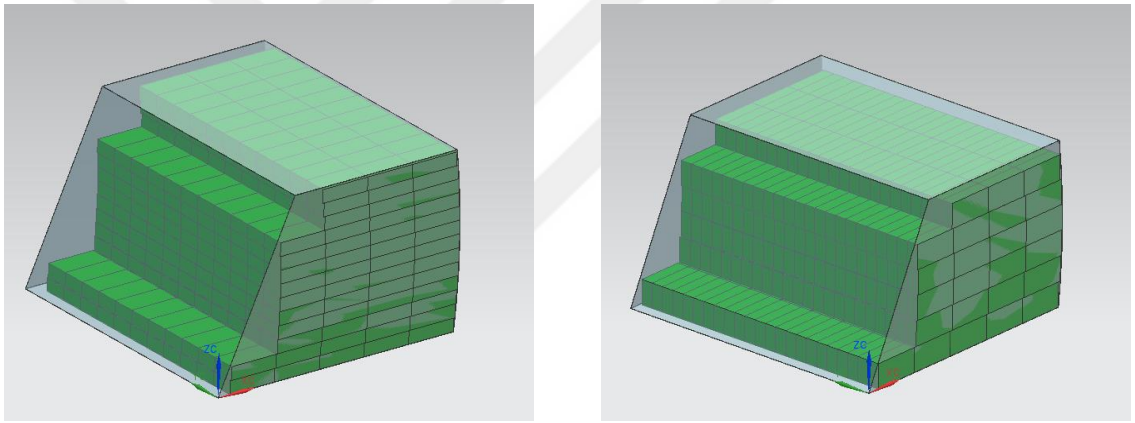
a) 5’inci yönde doldurma b) 6’ncı yönde doldurma

Şekil 4.5. Küboidlerin 5’inci ve 6’ncı yönlere göre düzlemsel sedan araç bagajı içine yerleştirilmesi

Düzlemsel yüzeyli sedan araç bagaj örneği için yapılan çalışma sonucunda elde edilen hacim değerleri karşılaştırılmış olup Şekil 4.5'te gösterilmiş olan 6'ıncı yön küboid için doldurulmuş bagaj örneğinde maksimum hacim değerine ulaşılmış olup belirlenen hacim değeri 540 litredir.

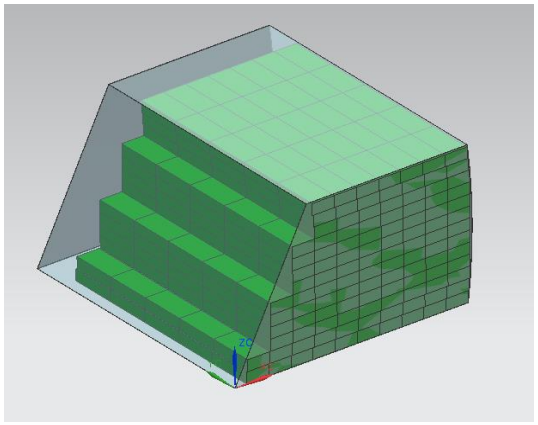
4.2. Ticari Araç Bagaj Hacmi

Ticari araç bagaj örneği için yapılan çalışma sonucunda elde edilen hacim değerleri karşılaştırılmış olup Şekil 4.6'da gösterilmiş olan 1'inci yön küboid ve Şekil 4.8'de gösterilmiş olan 6'ıncı yön küboid için doldurulmuş bagaj örneğinde maksimum hacim değerine ulaşılmış olup belirlenen hacim değeri 561 litredir. Şekil 4.7'de 3'üncü ve 4'üncü yönler için doldurulmuş bagaj çalışması ayrıca verilmiştir.

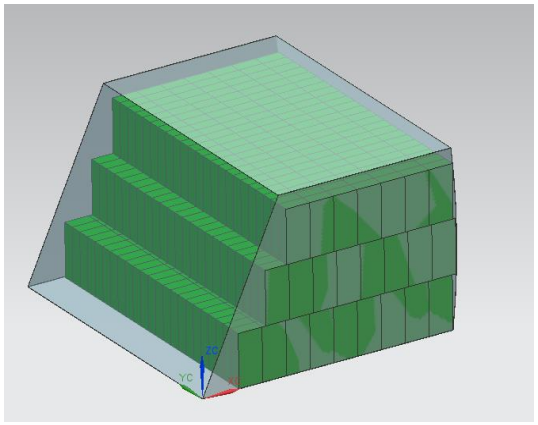


a) 1'inci yönde doldurma b) 2'nci yönde doldurma

Şekil 4.6. Küboid 1'inci yön ve 2'nci yön için doldurulmuş ticari araç bagaj örneği

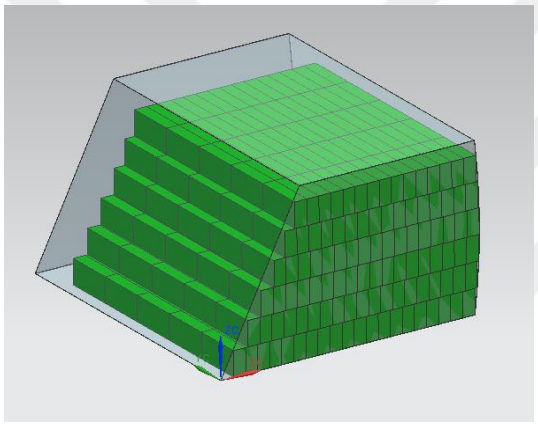


a) 3’üncü yönde doldurma

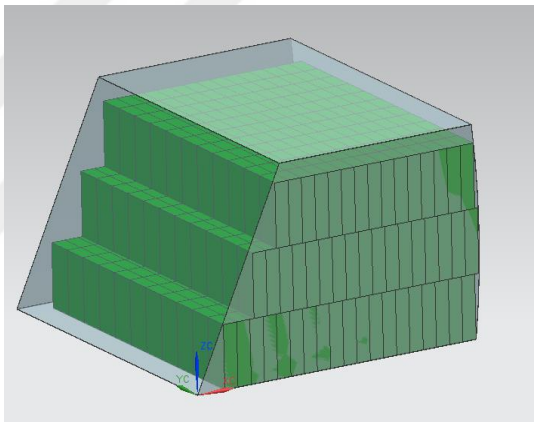


b) 4’üncü yönde doldurma

Şekil 4.7. Küboid 3’üncü yön ve 4’üncü yön için doldurulmuş ticari araç bagaj örneği



a) 5’inci yönde doldurma

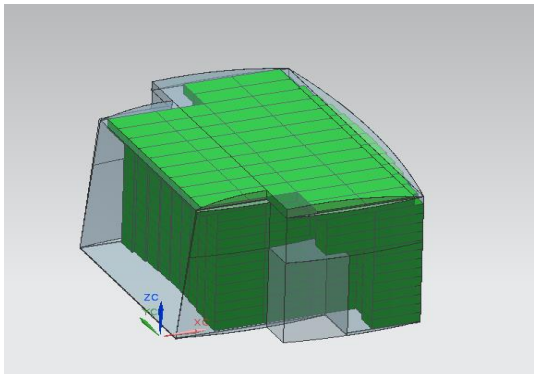


b) 6’ncı yönde doldurma

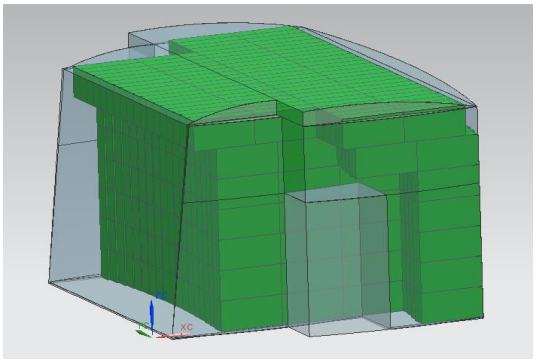
Şekil 4.8. Küboid 5’inci yön ve 6’ncı yön için doldurulmuş ticari araç bagaj örneği

4.3. Eğrisel Yüzeyli Sedan Araç Bagaj Hacmi

Eğrisel yüzeyli sedan araç bagaj örneği için yapılan çalışma sonucunda elde edilen hacim değerleri karşılaştırılmış olup Şekil 4.11’de gösterilmiş olan 5’inci yön küboid için doldurulmuş bagaj örneğinde maksimum hacim değerine ulaşılmış olup belirlenen hacim değeri 608 litredir. Diğer farklı yönler için elde edilen doldurulmuş bagaj hacim görüntüleri Şekil 4.9 ve Şekil 4.10’da verilmiştir.

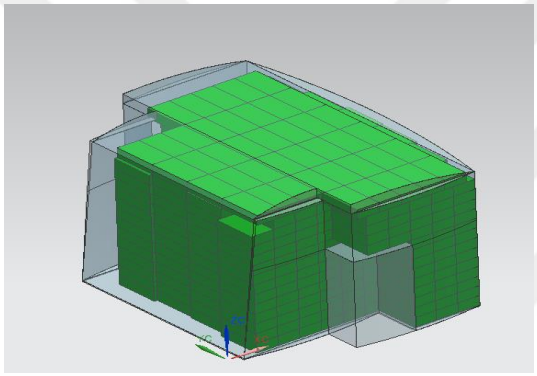


a) 1’inci yönde doldurma

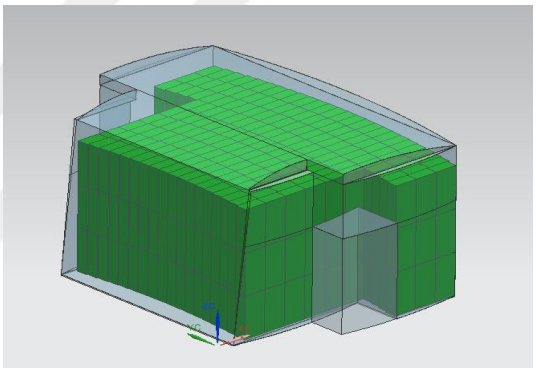


b) 2’nci yönde doldurma

Şekil 4.9. Küboid 1’inci yön ve 2’nci yön için doldurulmuş eğrisel yüzeyli sedan araç bagaj örneği

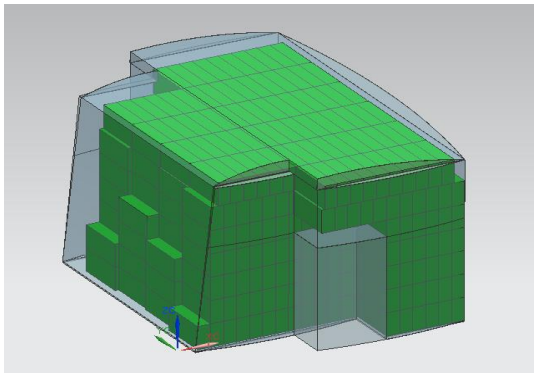


a) 3’üncü yönde doldurma

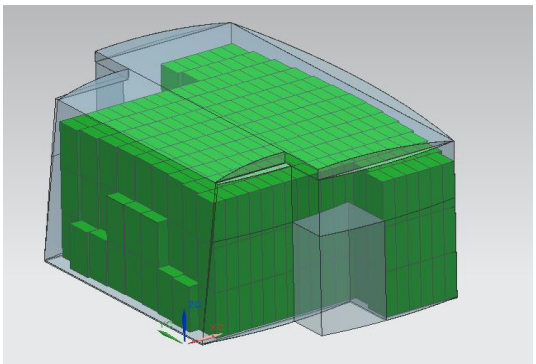


b) 4’üncü yönde doldurma

Şekil 4.10. Küboid 3’üncü yön ve 4’üncü yön için doldurulmuş eğrisel yüzeyli sedan araç bagaj örneği



a) 5’inci yönde doldurma



b) 6’ncı yönde doldurma

Şekil 4.11. Küboid 5’inci yön ve 6’ncı yön için doldurulmuş eğrisel yüzeyli sedan araç bagaj örneği

4.4. Farklı Hacim Hesaplamaların Karşılaştırılması

Her üç bagaj tasarımı için oluşturulan algoritma çalıştırılmış olup elde edilen sonuçlar bagaj tasarımlarına göre kendi içlerinde karşılaştırılarak hacim değerleri tespit edilmiştir (Çizelge 4.1). Düzlemsel yüzeyli bagaj (bkz. Şekil 3.1) ile eğrisel yüzeyli bagaj (bkz. Şekil 3.2) tasarım olarak birbirlerine çok yakın olmasına rağmen her küboid yönleri için elde edilen hacim değerleri farklı çıkmıştır. Bu durumun ortaya çıkmasında ki ana sebep tasarımın eğriselliğinden dolayı oluşan X ve Z vektörü yönünde boş alanın artması ve blokların bu vektörler yönünde daha fazla yerleşim yapmasıdır.

Çizelge 4.1. Küboid yönlerine göre bagaj hacimleri

Küboid Yönü	Düzlemsel Yüzeyli Sedan Araç Bagaj Hacmi (litre)	Ticari Araç Bagaj Hacmi (litre)	Eğrisel Yüzeyli Sedan Araç Bagaj Hacmi (litre)
Yön 1	536	561	589
Yön 2	488	552	576
Yön 3	534	545	603
Yön 4	522	552	558
Yön 5	502	525	608
Yön 6	540	561	568

Bu çalışmada elde edilmesi hedeflenen bir diğer ise hacim hesaplama süresinin en az seviyeye indirgemektir. Kuşkusuz oluşturulan algoritma ile operatör tarafından manuel yapılan işlem süresi çok daha az sürelere indirilmiştir. Algoritmanın çalışma süresini etkileyen bazı kısıtlar bulunmaktadır. Bunlardan en önemlileri bilgisayar performansı ve bagaj tasarımının kısıtlarıdır. Günümüz şartlarında ortalama bir bilgisayar performansı ile çalıştırılan bu algoritma ortalama 350-420 saniye civarında sürmektedir. Bu sürenin en önemli pay sahibi ise kesişim kontrolüdür. Otomotiv üreticilerinin bagaj içerisinde kullanmış oldukları kaplamanın sıkıştırılabilirlik mesafesi göz önüne alınarak kesişim kontrolleri blokların 1 mm hareketinden sonra tekrarlanmaktadır. Hareket mesafesenin arttırılması veya azaltılması algoritma çalışma süresine etki etmektedir. Bagaj tasarımı

içerisinde bulunabilecek araca ait farklı parçalar algoritma içerisinde daha fazla kesişim kontrolü yapılmasına ve algoritma çalışma süresinin uzamasına neden olacaktır. Yapmış olduğumuz çalışmada da algoritmanın en fazla çalışma süresine sahip olan bagaj tasarımı eğrisel yüzeyli sedan araç bagaj örneğidir.

5. SONUÇ

Araç üreticileri için önemli bir süreç olan bagaj hacimlerinin belirlenmesi için bu çalışmada geliştirilen bir algoritma ve bir uygulama sunulmuştur. Mevcut durumda araç üreticileri tasarım sürcinde bagaj hacminin hesaplanması işlemlerinin CAD ortamında manuel olarak kendi deneyimlerine göre yapmaktadırlar. Bu hesaplamanın CAD ortamında elle yapılması hatalara, eksik bagaj hacmi hesaplanmasına ve sürecin uzun zaman almasına neden olabilmektedir. Ayrıca araç tasarım sürecinde sıklıkla bagaj geometrileri değişebilmektedir. Bu da bagaj hacminin tekrar tekrar manuel olarak hesaplanması anlamına gelmektedir. Bu durum araç tasarım sürecini uzatmaktadır. Bu çalışma sonucunda geliştirilen algoritmanın, bagaj hacminin belirlenmesinde standartlarda yer alan kurallara göre doldurulması araç üreticilerine zaman ve maliyet açısından faydalı olacaktır.

Aynı araca ait tasarımlarda yapılan bazı değişimlerde algoritmanın temel amacı olan en az boşluk kalacak şekilde bagajı doldurma işleminin doğru çalıştığı görülmüştür. Sadece bazı yüzeylere gerçek bagaj tasarımına yakın olması amacıyla form verilmiş ve bundan doğan boşluğu algoritmanın algılamasıyla oluşan boşluklara da blok yerleştirmesi algoritmanın faydalı olduğunu kanıtlamıştır. Elde edilen hacim değerlerinden de bu durum gösterilmiştir.

Çalışmanın devamında bagaj yüzeyleri araç üreticilerinin tasarlamış olduğu gerçek modeller olacak şekilde çalışma genişletilebilir. Ayrıca, 6 farklı yönelimin karışık olarak kullanılabileceği durum, bir global optimizasyon algoritması ile kodlanarak bagaj hacmi içine maksimum sayıda bloğun yerleştirilmesi mümkün olabilir.

KAYNAKLAR

Başkan, Ö., Ceylan, H. 2014. Ulaşım ağ tasarımı problemlerinin çözümünde diferansiyel gelişim algoritması tabanlı çözüm yaklaşımları. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 20(9): 324-331.

Çalışır, K. 2015. Olimpik havuz plan şeması tasarımında genetik algoritmaya dayalı bir model. *Yüksek Lisans Tezi*, İTÜ Fen Bilimleri Enstitüsü, Bilişim Anabilim Dalı, İstanbul.

Dereli, T., Daş, G.S. 2010. Konteynır yükleme problemleri için karınca kolonisi optimizasyon yaklaşımı. *Gazi Üniv. Müh. Mim. Fak. Der*., 25(4): 881-894.

Domingo, B.M., Ponnambalam, S.G., Kanagaraj, G. 2013. A differential evolution based algorithm for single container loading problem. 2013 IEEE Symposium on Differential Evolution, 16-19 April 2013, Singapore.

Dziegielewski, A.V., Erbes, R. 2016. Fully automatic determination of the trunk volume. ATZlive Conferences for Vehicle and Engine Specialists, February 2016. Wiesbaden, Germany.

Gürbüz, M. Z., Akyokuş, S., Emiroğlu, İ., Güran A. 2009. An efficient algorithm for 3D rectangular box packing. Applied Automatic Systems: Proceedings of Selected AAS, 26-29 September 2009, Ohrid.

Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, USA, 232 pp.

International Standard, ISO 3832, 2002. Passenger cars - Luggage compartments - Method of measuring reference volume, Third edition.

Joung, Y., Noh S. 2014. Intelligent 3D packing using a grouping algorithm for automotive container engineering. *Journal of Computational Design and Engineering*, Vol. 1, No. 2: 140-151.

Kalınlı, A. 2012. Mühendislikte zeki programlama teknikleri ve uygulamaları. Erciyes Üniversitesi, Ders Notları, Kayseri, 154 s.

Karaboğa, D. 2004. Yapay zeka optimizasyon algoritmaları. Atlas Yayın Dağıtım, İstanbul, 246 s.

Kaymaz, İ. 2018. Optimizasyon teknikleri. Atatürk Üniversitesi, Mühendislik Fakültesi, Ders Notları, Erzurum, 29 s.

Liu H., Cai Z., Wang Y. 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, 10(2): 629-640.

Murty, K. G. 2003. Optimization Models For Decision Making: vol. 1. Internet Edition, USA, 598 pp.

Mutluer, M. 2007. Asenkron motor elektriksel eşdeğer devre parametrelerinin hibrid genetik algoritma yöntemiyle belirlenmesi. *Yüksek Lisans Tez,* SÜ Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği, Konya.

Özkan, R. 2003. Tek modelli deterministik montaj hattı dengeleme problemlerine genetik algoritma ile çözüm yaklaşımı. *Yüksek Lisans Tezi,* İTÜ Fen Bilimleri Enstitüsü, Endüstri Mühendisliği, İstanbul.

Özyön, S., Yaşar, C., Temurtas, H. 2011. Diferansiyel gelişim algoritmasının valf nokta etkili konveks olmayan ekonomik güç dağıtım problemlerine uygulanması. 6 th International Advanced Technologies Symposium, 16-18 May 2011, Elazığ, Türkiye.

Price, K., Storn, R. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, vol. 11, no. 4: 341–359.

SAE International, 2011. Interpretation of SAE J1100 Cargo Volume Indices, 01-0779.

Shellshear, E., Bohlin, R., Carlson, J. 2012. A combinatorial packing algorithm and standard trunk geometry for ISO luggage packing. Proceedings of ASME IDETC/CIE, Chicago, Illinois.

Shellshear, E., Bohlin, R., Carlson, J., Tafuri, S. 2015. A multi-threaded memetic packing algorithm for the ISO luggage packing problem. 2015 IEEE International Conference on Automation Science and Engineering, 24-28 August 2015, Gothenburg, Sweden.

Şeker, T. 2008. Düzlemsel çelik çerçevelerin genetik algoritma ile optimizasyonu. *Yüksek Lisans Tezi,* İTÜ Fen Bilimleri Enstitüsü, İnşaat Mühendisliği, İstanbul..

Tiwari, S., Fadel, G., Fenyes, P. 2010 A Fast and efficient compact packing algorithm for SAE and ISO luggage packing problems. *Journal of Computing and Information Science in Engineering*, vol. 10, no. 2.

EKLER

EK 1 Düzlemsel yüzeyli sedan araç bagaj hacmi hesaplama algoritması

EK 2 Ticari araç bagaj hacmi hesaplama algoritması

EK 3 Eğrisel yüzeyli sedan araç bagaj hacmi hesaplama algoritması

EK 1

```
ImportsSystem
ImportsNXOpen
ModuleNXJournal
Sub Main (ByValargs() As String)
Dim theSession As NXOpen.Session = NXOpen.Session.GetSession()
Dim workPart As NXOpen.Part = theSession.Parts.Work
Dim displayPart As NXOpen.Part = theSession.Parts.Display
Dim X, Y, Z As integer
X = 732
Y = -189
Z = 0
Dim basePoint1 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
Dim a, b, c As integer
a = 0
b = 0
c = 0
Dim basePoint3 As NXOpen.Point3d = New NXOpen.Point3d(a, b, c)
Dim orientation1 As NXOpen.Matrix3x3 = Nothing
orientation1.Xx = 1.0
orientation1.Xy = 0.0
orientation1.Xz = 0.0
orientation1.Yx = 0.0
orientation1.Yy = 1.0
orientation1.Yz = 0.0
orientation1.Zx = 0.0
orientation1.Zy = 0.0
orientation1.Zz = 1.0
Dim partLoadStatus1 As NXOpen.PartLoadStatus = Nothing
Dim component1 As NXOpen.Assemblies.Component = Nothing
component1 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\son_bagaj_model
i\montaj.prt", "MODEL", "basit", basePoint3, orientation1, -1,
partLoadStatus1, True)
Dim component2 As NXOpen.Assemblies.Component = Nothing
component2 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint1, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference11 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference11 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference11.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference11.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body22 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference11.FirstBody.Value = body22
Dim component122 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ön_yüzey 1"),
NXOpen.Assemblies.Component)
Dim body122 As NXOpen.Body =
CType(component122.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference11.SecondBody.Value = body122
Dim cy As Integer
cy = 2
Dim result11 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject11 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Do While (result11 = cy)
Dim pt22 As Vector3d
pt22.X = -1
pt22.Y = 0
pt22.Z = 0
workPart.ComponentAssembly.MoveComponent(component2, pt22, orientation1)
Dim result22 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject22 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Loop
simpleInterference11.Destroy()
Dim i, f, n As Integer
For n = 0 To 100
Dim component3 As NXOpen.Assemblies.Component = Nothing
For f = 0 To 100
For i = 0 To 100
X = X - 200
Dim basePoint2 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint2, orientation1, -1, partLoadStatus1, True)
```

```
Dim simpleInterference1 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference1 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference1.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference1.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body1 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.FirstBody.Value = body1
Dim body2 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.SecondBody.Value = body2
Dim t As Integer
t = 2
Dim result1 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject1 As NXOpen.NXObject = Nothing
nXObject1 = simpleInterference1.Commit()
Do While (result1 = t)
Dim pt2 As Vector3d
pt2.X = -1
pt2.Y = 0
pt2.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt2, orientation1)
Dim result2 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject2 As NXOpen.NXObject = Nothing
nXObject2 = simpleInterference1.Commit()
Loop
simpleInterference1.Destroy()
Dim simpleInterference2 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference2 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference2.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
```

```
simpleInterference2.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body3 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference2.FirstBody.Value = body3
Dim component23 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT arka_yüzey 1"),
NXOpen.Assemblies.Component)
Dim body23 As NXOpen.Body =
CType(component23.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference2.SecondBody.Value = body23
Dim k As Integer
k = 2
Dim result3 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result3 = simpleInterference2.PerformCheck()
Dim nXObject3 As NXOpen.NXObject = Nothing
nXObject3 = simpleInterference2.Commit()
simpleInterference2.Destroy()
If result3 = k Then
Dim markId2 As NXOpen.Session.UndoMarkId = Nothing
markId2 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs1 As Integer = Nothing
nErrs1 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs2 As Integer = Nothing
nErrs2 = theSession.UpdateManager.DoUpdate(markId2)
ExitFor
EndIf
Dim simpleInterference27 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference27 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference27.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference27.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference27.FirstBody.Value = body3
Dim component24 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_1 1"),
NXOpen.Assemblies.Component)
```

```
Dim body24 As NXOpen.Body =
CType(component24.FindObject("PARTIAL_PROTO#.Bodies|Body10"), NXOpen.Body)
simpleInterference27.SecondBody.Value = body24
Dim ka As Integer
ka = 2
Dim result37 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result37 = simpleInterference27.PerformCheck()
Dim nXObject37 As NXOpen.NXObject = Nothing
nXObject37 = simpleInterference27.Commit()
simpleInterference27.Destroy()
If result37 = kaThen
Dim markId27 As NXOpen.Session.UndoMarkId = Nothing
markId27 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs17 As Integer = Nothing
nErrs17 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs27 As Integer = Nothing
nErrs27 = theSession.UpdateManager.DoUpdate(markId27)
ExitFor
EndIf
Dim simpleInterference28 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference28 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference28.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference28.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference28.FirstBody.Value = body3
Dim component25 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_1 1"),
NXOpen.Assemblies.Component)
Dim body25 As NXOpen.Body =
CType(component25.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference28.SecondBody.Value = body25
Dim kb As Integer
kb = 2
Dim result38 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result38 = simpleInterference28.PerformCheck()
Dim nXObject38 As NXOpen.NXObject = Nothing
nXObject38 = simpleInterference28.Commit()
```

```
simpleInterference28.Destroy()
If result38 = kbThen
Dim markId28 As NXOpen.Session.UndoMarkId = Nothing
markId28 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs18 As Integer = Nothing
nErrs18 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs28 As Integer = Nothing
nErrs28 = theSession.UpdateManager.DoUpdate(markId28)
ExitFor
EndIf
Dim simpleInterference29 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference29 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference29.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference29.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference29.FirstBody.Value = body3
Dim component26 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_2 1"),
NXOpen.Assemblies.Component)
Dim body26 As NXOpen.Body =
CType(component26.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference29.SecondBody.Value = body26
Dim kc As Integer
kc = 2
Dim result39 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result39 = simpleInterference29.PerformCheck()
Dim nXObject39 As NXOpen.NXObject = Nothing
nXObject39 = simpleInterference29.Commit()
simpleInterference29.Destroy()
If result39 = kcThen
Dim markId29 As NXOpen.Session.UndoMarkId = Nothing
markId29 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs19 As Integer = Nothing
nErrs19 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs29 As Integer = Nothing
nErrs29 = theSession.UpdateManager.DoUpdate(markId29)
```

```
ExitFor
EndIf
Dim simpleInterference59 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference59 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference59.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference59.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference59.FirstBody.Value = body3
Dim component56 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_2 1"),
NXOpen.Assemblies.Component)
Dim body56 As NXOpen.Body =
CType(component56.FindObject("PARTIAL_PROTO#.Bodies|Body10"), NXOpen.Body)
simpleInterference59.SecondBody.Value = body56
Dim kd As Integer
kd = 2
Dim result59 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result59 = simpleInterference59.PerformCheck()
Dim nXObject59 As NXOpen.NXObject = Nothing
nXObject59 = simpleInterference59.Commit()
simpleInterference59.Destroy()
If result59 = kdThen
Dim markId59 As NXOpen.Session.UndoMarkId = Nothing
markId59 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs59 As Integer = Nothing
 nErrs59 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs69 As Integer = Nothing
nErrs69 = theSession.UpdateManager.DoUpdate(markId59)
ExitFor
EndIf
component2 = component3
Next i
 X = 732
Y = 100*(f+1)-190
Dim basePoint6 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
```

```
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint6, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference3 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference3 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference3.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference3.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body6 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference3.FirstBody.Value = body6
Dim component13 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_2 1"),
NXOpen.Assemblies.Component)
Dim body5 As NXOpen.Body =
CType(component13.FindObject("PARTIAL_PROTO#.Bodies|Body11"), NXOpen.Body)
simpleInterference3.SecondBody.Value = body5
Dim m As Integer
m = 2
Dim result4 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result4 = simpleInterference3.PerformCheck()
Dim nXObject4 As NXOpen.NXObject = Nothing
nXObject4 = simpleInterference3.Commit()
simpleInterference3.Destroy()
If result4 = m Then
Dim markId3 As NXOpen.Session.UndoMarkId = Nothing
markId3 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs3 As Integer = Nothing
nErrs3 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.DoUpdate(markId3)
ExitFor
EndIf
Dim simpleInterference33 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference33 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference33.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference33.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body66 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference33.FirstBody.Value = body66
Dim body55 As NXOpen.Body =
CType(component13.FindObject("PARTIAL_PROTO#.Bodies|Body10"), NXOpen.Body)
simpleInterference33.SecondBody.Value = body55
Dim mt As Integer
mt = 2
Dim result44 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result44 = simpleInterference33.PerformCheck()
Dim nXObject44 As NXOpen.NXObject = Nothing
nXObject44 = simpleInterference33.Commit()
simpleInterference33.Destroy()
If result44 = mtThen
Dim markId33 As NXOpen.Session.UndoMarkId = Nothing
markId33 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs33 As Integer = Nothing
nErrs33 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs44 As Integer = Nothing
nErrs44 = theSession.UpdateManager.DoUpdate(markId33)
ExitFor
EndIf
component2=component3
Next f
Y = -189
X = 732
Z = 50*(n+1)
Dim basePoint7 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
Dim component7 As NXOpen.Assemblies.Component = Nothing
component7 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID (i)", basePoint7, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference4 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference4 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference4.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference4.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body8 As NXOpen.Body =
CType(component7.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference4.FirstBody.Value = body8
Dim component14 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT üst_yüzey 1"),
NXOpen.Assemblies.Component)
Dim body9 As NXOpen.Body =
CType(component14.FindObject("PARTIAL_PROTO#.Bodies|Body11"), NXOpen.Body)
simpleInterference4.SecondBody.Value = body9
Dim g As Integer
g = 2
Dim result5 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result5 = simpleInterference4.PerformCheck()
Dim nXObject5 As NXOpen.NXObject = Nothing
nXObject5 = simpleInterference4.Commit()
simpleInterference4.Destroy()
If result5 = g Then
Dim markId4 As NXOpen.Session.UndoMarkId = Nothing
markId4 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.AddToDeleteList(COMPONENT7)
Dim nErrs5 As Integer = Nothing
nErrs5 = theSession.UpdateManager.DoUpdate(markId4)
ExitFor
EndIf
Next n
partLoadStatus1.Dispose()
Dim objects1(-1) As NXOpen.NXObject
EndSub
EndModule
```

EK 2

```
ImportsSystem
ImportsNXOpen
ModuleNXJournal
Sub Main (ByValargs() As String)
Dim theSession As NXOpen.Session = NXOpen.Session.GetSession()
Dim workPart As NXOpen.Part = theSession.Parts.Work
Dim displayPart As NXOpen.Part = theSession.Parts.Display
Dim X, Y, Z As integer
X = 873
Y = 0
Z = 0
Dim basePoint1 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
Dim a, b, c As integer
a = 0
b = 0
c = 0
Dim basePoint3 As NXOpen.Point3d = New NXOpen.Point3d(a, b, c)
Dim orientation1 As NXOpen.Matrix3x3 = Nothing
orientation1.Xx = 1.0
orientation1.Xy = 0.0
orientation1.Xz = 0.0
orientation1.Yx = 0.0
orientation1.Yy = 1.0
orientation1.Yz = 0.0
orientation1.Zx = 0.0
orientation1.Zy = 0.0
orientation1.Zz = 1.0
Dim partLoadStatus1 As NXOpen.PartLoadStatus = Nothing
Dim component1 As NXOpen.Assemblies.Component = Nothing
component1 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\ticari\ticari_m
ontaj.prt", "MODEL", "basit", basePoint3, orientation1, -1, partLoadStatus1,
True)
Dim component2 As NXOpen.Assemblies.Component = Nothing
component2 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint1, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference11 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference11 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference11.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference11.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body22 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference11.FirstBody.Value = body22
Dim component122 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ticari_arka 1"),
NXOpen.Assemblies.Component)
Dim body122 As NXOpen.Body =
CType(component122.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference11.SecondBody.Value = body122
Dim cy As Integer
cy = 2
Dim result11 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject11 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Do While (result11 = cy)
Dim pt22 As Vector3d
pt22.X = -1
pt22.Y = 0
pt22.Z = 0
workPart.ComponentAssembly.MoveComponent(component2, pt22, orientation1)
Dim result22 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject22 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Loop
simpleInterference11.Destroy()
Dim i, f, n As Integer
For n = 0 To 100
Dim component3 As NXOpen.Assemblies.Component = Nothing
For f = 0 To 100
For i = 0 To 100
X = X - 200
Dim basePoint2 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint2, orientation1, -1, partLoadStatus1, True)
```

```
Dim simpleInterference1 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference1 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference1.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference1.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body1 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.FirstBody.Value = body1
Dim body2 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.SecondBody.Value = body2
Dim t As Integer
t = 2
Dim result1 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject1 As NXOpen.NXObject = Nothing
nXObject1 = simpleInterference1.Commit()
Do While (result1 = t)
Dim pt2 As Vector3d
pt2.X = -1
pt2.Y = 0
pt2.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt2, orientation1)
Dim result2 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject2 As NXOpen.NXObject = Nothing
nXObject2 = simpleInterference1.Commit()
Loop
simpleInterference1.Destroy()
Dim simpleInterference2 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference2 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference2.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
```

```
simpleInterference2.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body3 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference2.FirstBody.Value = body3
Dim component23 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ticari_ön 1"),
NXOpen.Assemblies.Component)
Dim body23 As NXOpen.Body =
CType(component23.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference2.SecondBody.Value = body23
Dim k As Integer
k = 2
Dim result3 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result3 = simpleInterference2.PerformCheck()
Dim nXObject3 As NXOpen.NXObject = Nothing
nXObject3 = simpleInterference2.Commit()
simpleInterference2.Destroy()
If result3 = k Then
Dim markId2 As NXOpen.Session.UndoMarkId = Nothing
markId2 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs1 As Integer = Nothing
nErrs1 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs2 As Integer = Nothing
nErrs2 = theSession.UpdateManager.DoUpdate(markId2)
ExitFor
EndIf
component2 = component3
Next i
X = 873
Y = 100 * (f + 1)
Dim basePoint6 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint6, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference3 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference3 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference3.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference3.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body6 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference3.FirstBody.Value = body6
Dim component13 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ticari_yan_2 1"),
NXOpen.Assemblies.Component)
Dim body5 As NXOpen.Body =
CType(component13.FindObject("PARTIAL_PROTO#.Bodies|Body11"), NXOpen.Body)
simpleInterference3.SecondBody.Value = body5
Dim m As Integer
m = 2
Dim result4 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result4 = simpleInterference3.PerformCheck()
Dim nXObject4 As NXOpen.NXObject = Nothing
nXObject4 = simpleInterference3.Commit()
simpleInterference3.Destroy()
If result4 = m Then
Dim markId3 As NXOpen.Session.UndoMarkId = Nothing
markId3 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs3 As Integer = Nothing
nErrs3 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.DoUpdate(markId3)
ExitFor
EndIf
Dim simpleInterference111 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference111 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference111.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference111.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference111.FirstBody.Value = body6
```

```
simpleInterference111.SecondBody.Value = body122
Dim sy As Integer
sy = 2
Dim result111 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result111 = simpleInterference111.PerformCheck()
Dim nXObject111 As NXOpen.NXObject = Nothing
nXObject111 = simpleInterference111.Commit()
Do While (result111 = sy)
Dim pt222 As Vector3d
pt222.X = -1
pt222.Y = 0
pt222.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt222, orientation1)
Dim result222 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result111 = simpleInterference111.PerformCheck()
Dim nXObject222 As NXOpen.NXObject = Nothing
nXObject111 = simpleInterference111.Commit()
Loop
simpleInterference111.Destroy()
component2=component3
Next f
Y = 0
X = 873
Z = 50*(n+1)
Dim basePoint7 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
Dim component7 As NXOpen.Assemblies.Component = Nothing
component7 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID (i)", basePoint7, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference4 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference4 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference4.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference4.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body8 As NXOpen.Body =
CType(component7.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference4.FirstBody.Value = body8
```

```
Dim component14 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ticari_üst 1"),
NXOpen.Assemblies.Component)
Dim body9 As NXOpen.Body =
CType(component14.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference4.SecondBody.Value = body9
Dim g As Integer
g = 2
Dim result5 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result5 = simpleInterference4.PerformCheck()
Dim nXObject5 As NXOpen.NXObject = Nothing
nXObject5 = simpleInterference4.Commit()
simpleInterference4.Destroy()
If result5 = g Then
Dim markId4 As NXOpen.Session.UndoMarkId = Nothing
markId4 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.AddToDeleteList(COMPONENT7)
Dim nErrs5 As Integer = Nothing
nErrs5 = theSession.UpdateManager.DoUpdate(markId4)
ExitFor
EndIf
Dim simpleInterference1111 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference1111 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference1111.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference1111.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference1111.FirstBody.Value = body8
simpleInterference1111.SecondBody.Value = body122
Dim py As Integer
py = 2
Dim result1111 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1111 = simpleInterference1111.PerformCheck()
Dim nXObject1111 As NXOpen.NXObject = Nothing
nXObject1111 = simpleInterference1111.Commit()
Do While (result1111 = py)
Dim pt2222 As Vector3d
```

```
pt2222.X = -1
pt2222.Y = 0
pt2222.Z = 0
workPart.ComponentAssembly.MoveComponent(component7, pt2222, orientation1)
Dim result2222 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1111 = simpleInterference1111.PerformCheck()
Dim nXObject2222 As NXOpen.NXObject = Nothing
nXObject1111 = simpleInterference1111.Commit()
Loop
simpleInterference1111.Destroy()
component2=component7
Next n
partLoadStatus1.Dispose()
Dim objects1(-1) As NXOpen.NXObject
EndSub
EndModule
```

EK 3

```
ImportsSystem
ImportsNXOpen
ModuleNXJournal
Sub Main(ByValargs() As String)
Dim theSession As NXOpen.Session = NXOpen.Session.GetSession()
Dim workPart As NXOpen.Part = theSession.Parts.Work
Dim displayPart As NXOpen.Part = theSession.Parts.Display
Dim X, Y, Z As Integer
X = 782
Y = -222
Z = 0
Dim basePoint1 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
Dim a, b, c As Integer
a = 0
b = 0
c = 0
Dim basePoint3 As NXOpen.Point3d = New NXOpen.Point3d(a, b, c)
Dim orientation1 As NXOpen.Matrix3x3 = Nothing
orientation1.Xx = 1.0
orientation1.Xy = 0.0
orientation1.Xz = 0.0
orientation1.Yx = 0.0
orientation1.Yy = 1.0
orientation1.Yz = 0.0
orientation1.Zx = 0.0
orientation1.Zy = 0.0
orientation1.Zz = 1.0
Dim partLoadStatus1 As NXOpen.PartLoadStatus = Nothing
Dim component1 As NXOpen.Assemblies.Component = Nothing
component1 =
workPart.ComponentAssembly.AddComponent("C:\son_bagaj_modeli\montaj_eğri.prt",
"MODEL", "basit", basePoint3, orientation1, -1, partLoadStatus1, True)
Dim component2 As NXOpen.Assemblies.Component = Nothing
component2 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint1, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference11 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference11 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference11.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference11.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body22 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference11.FirstBody.Value = body22
Dim component122 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ön_yüzey_eğri 1"),
NXOpen.Assemblies.Component)
Dim body122 As NXOpen.Body =
CType(component122.FindObject("PARTIAL_PROTO#.Bodies|Body10"), NXOpen.Body)
simpleInterference11.SecondBody.Value = body122
Dim cy As Integer
cy = 2
Dim result11 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject11 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Do While (result11 = cy)
Dim pt22 As Vector3d
pt22.X = -1
pt22.Y = 0
pt22.Z = 0
workPart.ComponentAssembly.MoveComponent(component2, pt22, orientation1)
Dim result22 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result11 = simpleInterference11.PerformCheck()
Dim nXObject22 As NXOpen.NXObject = Nothing
nXObject11 = simpleInterference11.Commit()
Loop
simpleInterference11.Destroy()
Dim i, f, n, u As Integer
Dim sayac As Integer
sayac = 0
For n = 0 To 100
Dim component3 As NXOpen.Assemblies.Component = Nothing
For f = 0 To 100
For i = 0 To 100
X = X - 200 - sayac
Dim basePoint2 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
```

```
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint2, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference1 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference1 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference1.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference1.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body1 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.FirstBody.Value = body1
Dim body2 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference1.SecondBody.Value = body2
Dim t As Integer
t = 2
Dim result1 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject1 As NXOpen.NXObject = Nothing
nXObject1 = simpleInterference1.Commit()
Do While (result1 = t)
Dim pt2 As Vector3d
pt2.X = -1
pt2.Y = 0
pt2.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt2, orientation1)
Dim result2 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result1 = simpleInterference1.PerformCheck()
Dim nXObject2 As NXOpen.NXObject = Nothing
nXObject2 = simpleInterference1.Commit()
Loop
simpleInterference1.Destroy()
Dim simpleInterference190 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference190 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference190.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference190.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body190 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference190.FirstBody.Value = body190
Dim body290 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference190.SecondBody.Value = body290
Dim tg As Integer
tg = 0
Dim result190 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result190 = simpleInterference190.PerformCheck()
Dim nXObject190 As NXOpen.NXObject = Nothing
nXObject190 = simpleInterference190.Commit()
Do While (result190 = tg)
Dim pt290 As Vector3d
pt290.X = 1
pt290.Y = 0
pt290.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt290, orientation1)
Dim result290 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result190 = simpleInterference190.PerformCheck()
Dim nXObject290 As NXOpen.NXObject = Nothing
nXObject290 = simpleInterference190.Commit()
Loop
simpleInterference190.Destroy()
Dim simpleInterference2 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference2 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference2.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference2.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body3 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference2.FirstBody.Value = body3
```

```
Dim component23 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT arka_yüzey 1"),
NXOpen.Assemblies.Component)
Dim body23 As NXOpen.Body =
CType(component23.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference2.SecondBody.Value = body23
Dim k As Integer
k = 2
Dim result3 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result3 = simpleInterference2.PerformCheck()
Dim nXObject3 As NXOpen.NXObject = Nothing
nXObject3 = simpleInterference2.Commit()
simpleInterference2.Destroy()
If result3 = k Then
Dim markId2 As NXOpen.Session.UndoMarkId = Nothing
markId2 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs1 As Integer = Nothing
nErrs1 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs2 As Integer = Nothing
nErrs2 = theSession.UpdateManager.DoUpdate(markId2)
ExitFor
EndIf
Dim simpleInterference27 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference27 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference27.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference27.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference27.FirstBody.Value = body3
Dim component24 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_eğri_1 1"),
NXOpen.Assemblies.Component)
Dim body24 As NXOpen.Body =
CType(component24.FindObject("PARTIAL_PROTO#.Bodies|Body14"), NXOpen.Body)
simpleInterference27.SecondBody.Value = body24
Dim ka As Integer
ka = 2
Dim result37 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
```

```
result37 = simpleInterference27.PerformCheck()
Dim nXObject37 As NXOpen.NXObject = Nothing
nXObject37 = simpleInterference27.Commit()
simpleInterference27.Destroy()
If result37 = kaThen
Dim markId27 As NXOpen.Session.UndoMarkId = Nothing
markId27 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs17 As Integer = Nothing
nErrs17 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs27 As Integer = Nothing
nErrs27 = theSession.UpdateManager.DoUpdate(markId27)
ExitFor
EndIf
Dim simpleInterference28 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference28 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference28.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference28.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference28.FirstBody.Value = body3
Dim component25 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_eğri_1 1"),
NXOpen.Assemblies.Component)
Dim body25 As NXOpen.Body =
CType(component25.FindObject("PARTIAL_PROTO#.Bodies|Body16"), NXOpen.Body)
simpleInterference28.SecondBody.Value = body25
Dim kb As Integer
kb = 2
Dim result38 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result38 = simpleInterference28.PerformCheck()
Dim nXObject38 As NXOpen.NXObject = Nothing
nXObject38 = simpleInterference28.Commit()
simpleInterference28.Destroy()
If result38 = kbThen
Dim markId28 As NXOpen.Session.UndoMarkId = Nothing
markId28 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs18 As Integer = Nothing
```

```
nErrs18 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs28 As Integer = Nothing
nErrs28 = theSession.UpdateManager.DoUpdate(markId28)
ExitFor
EndIf
Dim simpleInterference29 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference29 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference29.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference29.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference29.FirstBody.Value = body3
Dim component26 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_eğri_2_2 1"),
NXOpen.Assemblies.Component)
Dim body26 As NXOpen.Body =
CType(component26.FindObject("PARTIAL_PROTO#.Bodies|Body23"), NXOpen.Body)
simpleInterference29.SecondBody.Value = body26
Dim kc As Integer
kc = 2
Dim result39 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result39 = simpleInterference29.PerformCheck()
Dim nXObject39 As NXOpen.NXObject = Nothing
nXObject39 = simpleInterference29.Commit()
simpleInterference29.Destroy()
If result39 = kcThen
Dim markId29 As NXOpen.Session.UndoMarkId = Nothing
markId29 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs19 As Integer = Nothing
nErrs19 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs29 As Integer = Nothing
nErrs29 = theSession.UpdateManager.DoUpdate(markId29)
ExitFor
EndIf
Dim simpleInterference59 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference59 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference59.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference59.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
simpleInterference59.FirstBody.Value = body3
Dim component56 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_eğri_2_2 1"),
NXOpen.Assemblies.Component)
Dim body56 As NXOpen.Body =
CType(component56.FindObject("PARTIAL_PROTO#.Bodies|Body25"), NXOpen.Body)
simpleInterference59.SecondBody.Value = body56
Dim kd As Integer
kd = 2
Dim result59 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result59 = simpleInterference59.PerformCheck()
Dim nXObject59 As NXOpen.NXObject = Nothing
nXObject59 = simpleInterference59.Commit()
simpleInterference59.Destroy()
If result59 = kdThen
Dim markId59 As NXOpen.Session.UndoMarkId = Nothing
markId59 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs59 As Integer = Nothing
nErrs59 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs69 As Integer = Nothing
nErrs69 = theSession.UpdateManager.DoUpdate(markId59)
ExitFor
EndIf
component2 = component3
sayac = 0
Next i
X = 782
Y = 100 * (f + 1) - 222
Dim basePoint6 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint6, orientation1, -1, partLoadStatus1, True)
Dim simpleInterference39 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference39 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
```

```
simpleInterference39.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference39.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body69 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference39.FirstBody.Value = body69
Dim component139 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT ön_yüzey_eğri 1"),
NXOpen.Assemblies.Component)
Dim body59 As NXOpen.Body =
CType(component139.FindObject("PARTIAL_PROTO#.Bodies|Body10"), NXOpen.Body)
simpleInterference39.SecondBody.Value = body59
Dim mg As Integer
mg = 0
Dim result49 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result49 = simpleInterference39.PerformCheck()
Dim nXObject49 As NXOpen.NXObject = Nothing
nXObject49 = simpleInterference39.Commit()
Do While (result49 = mg)
Dim pt29 As Vector3d
pt29.X = 1
pt29.Y = 0
pt29.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt29, orientation1)
Dim result499 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result49 = simpleInterference39.PerformCheck()
Dim nXObject499 As NXOpen.NXObject = Nothing
nXObject499 = simpleInterference39.Commit()
Loop
simpleInterference39.Destroy()
Dim p As Integer
For p=0 to 1000
Dim simpleInterference489 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference489 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference489.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
```

```
simpleInterference489.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body88 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference489.FirstBody.Value = body88
Dim component148 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT üst_yüzey_eğri_2 1"),
NXOpen.Assemblies.Component)
Dim body98 As NXOpen.Body =
CType(component148.FindObject("PARTIAL_PROTO#.Bodies|Body16"), NXOpen.Body)
simpleInterference489.SecondBody.Value = body98
Dim gf As Integer
gf = 2
Dim result598 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result598 = simpleInterference489.PerformCheck()
Dim nXObject598 As NXOpen.NXObject = Nothing
nXObject598 = simpleInterference489.Commit()
If result598 = gfThen
Dim pt232 As Vector3d
pt232.X = -1
pt232.Y = 0
pt232.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt232, orientation1)
Dim result232 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result598 = simpleInterference489.PerformCheck()
Dim nXObject232 As NXOpen.NXObject = Nothing
nXObject232 = simpleInterference489.Commit()
Else
ExitFor
EndIf
sayac = sayac + 1
simpleInterference489.Destroy()
next p
Dim simpleInterference3 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference3 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference3.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
```

```
simpleInterference3.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body6 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference3.FirstBody.Value = body6
Dim component13 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT yan_yüzey_eğri_2_2 1"),
NXOpen.Assemblies.Component)
Dim body5 As NXOpen.Body =
CType(component13.FindObject("PARTIAL_PROTO#.Bodies|Body27"), NXOpen.Body)
simpleInterference3.SecondBody.Value = body5
Dim m As Integer
m = 2
Dim result4 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result4 = simpleInterference3.PerformCheck()
Dim nXObject4 As NXOpen.NXObject = Nothing
nXObject4 = simpleInterference3.Commit()
simpleInterference3.Destroy()
If result4 = m Then
Dim markId3 As NXOpen.Session.UndoMarkId = Nothing
markId3 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs3 As Integer = Nothing
nErrs3 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.DoUpdate(markId3)
ExitFor
EndIf
Dim simpleInterference33 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference33 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference33.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference33.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body66 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference33.FirstBody.Value = body66
```

```
Dim body55 As NXOpen.Body =
CType(component13.FindObject("PARTIAL_PROTO#.Bodies|Body25"), NXOpen.Body)
simpleInterference33.SecondBody.Value = body55
Dim mt As Integer
mt = 2
Dim result44 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result44 = simpleInterference33.PerformCheck()
Dim nXObject44 As NXOpen.NXObject = Nothing
nXObject44 = simpleInterference33.Commit()
simpleInterference33.Destroy()
If result44 = mtThen
Dim markId33 As NXOpen.Session.UndoMarkId = Nothing
markId33 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs33 As Integer = Nothing
nErrs33 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs44 As Integer = Nothing
nErrs44 = theSession.UpdateManager.DoUpdate(markId33)
ExitFor
EndIf
component2 = component3
Next f
Y = -222
X = 782
Z = 50 * (n + 1)
Dim basePoint7 As NXOpen.Point3d = New NXOpen.Point3d(X, Y, Z)
component3 =
workPart.ComponentAssembly.AddComponent("C:\Users\Acer\Desktop\NX\cuboid.prt",
"MODEL", "CUBOID", basePoint7, orientation1, -1, partLoadStatus1, True)
For u=0 to 1000
Dim simpleInterference490 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference490 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference490.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference490.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body89 As NXOpen.Body =
CType(component2.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference490.FirstBody.Value = body89
```

```
Dim component149 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT üst_yüzey_eğri_2 1"),
NXOpen.Assemblies.Component)
Dim body99 As NXOpen.Body =
CType(component149.FindObject("PARTIAL_PROTO#.Bodies|Body16"), NXOpen.Body)
simpleInterference490.SecondBody.Value = body99
Dim gt As Integer
gt = 2
Dim result599 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result599 = simpleInterference490.PerformCheck()
Dim nXObject599 As NXOpen.NXObject = Nothing
nXObject599 = simpleInterference490.Commit()
If result599 = gtThen
Dim pt231 As Vector3d
pt231.X = -1
pt231.Y = 0
pt231.Z = 0
workPart.ComponentAssembly.MoveComponent(component3, pt231, orientation1)
Dim result231 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result599 = simpleInterference490.PerformCheck()
Dim nXObject231 As NXOpen.NXObject = Nothing
nXObject231 = simpleInterference490.Commit()
Else
ExitFor
EndIf
sayac = sayac + 1
simpleInterference490.Destroy()
next u
Dim simpleInterference4 As NXOpen.GeometricAnalysis.SimpleInterference =
Nothing
simpleInterference4 =
workPart.AnalysisManager.CreateSimpleInterferenceObject()
simpleInterference4.InterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.InterferenceMethod.InterferenceSol
id
simpleInterference4.FaceInterferenceType =
NXOpen.GeometricAnalysis.SimpleInterference.FaceInterferenceMethod.FirstPairOn
ly
Dim body8 As NXOpen.Body =
CType(component3.FindObject("PROTO#.Bodies|EXTRUDE(2)"), NXOpen.Body)
simpleInterference4.FirstBody.Value = body8
```

```
Dim component14 As NXOpen.Assemblies.Component =
CType(component1.FindObject("COMPONENT üst_sinir 1"),
NXOpen.Assemblies.Component)
Dim body9 As NXOpen.Body =
CType(component14.FindObject("PARTIAL_PROTO#.Bodies|Body9"), NXOpen.Body)
simpleInterference4.SecondBody.Value = body9
Dim g As Integer
g = 2
Dim result5 As NXOpen.GeometricAnalysis.SimpleInterference.Result = Nothing
result5 = simpleInterference4.PerformCheck()
Dim nXObject5 As NXOpen.NXObject = Nothing
nXObject5 = simpleInterference4.Commit()
simpleInterference4.Destroy()
If result5 = g Then
Dim markId4 As NXOpen.Session.UndoMarkId = Nothing
markId4 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible,
"Delete")
Dim nErrs4 As Integer = Nothing
nErrs4 = theSession.UpdateManager.AddToDeleteList(COMPONENT3)
Dim nErrs5 As Integer = Nothing
nErrs5 = theSession.UpdateManager.DoUpdate(markId4)
ExitFor
EndIf
component2 = component3
Next n
partLoadStatus1.Dispose()
Dim objects1(-1) As NXOpen.NXObject
EndSub
EndModule
```

ÖZGEÇMİŞ

Adı Soyadı	: İbrahim Halil DEMİR
Doğum Yeri ve Tarihi	: Batman, 11.10.1991
Yabancı Dil	: İngilizce

Eğitim Durumu

Lise	: Cengizhan Anadolu Lisesi, 2005-2009
Lisans	: Uludağ Üniversitesi, Makine Mühendisliği Bölümü, 2009-2013
Yüksek Lisans	: Uludağ Üniversitesi, Fen Bilimleri Enstitüsü, Makine Mühendisliği Anabilim Dalı, 2015-2019

Çalıştığı Kurum/Kurumlar	: İşkar Makina, 2015-2016 Bahadır Teknik/ Mitsubishi Electric, 2016-2017 Karayolları Genel Müdürlüğü, 2017-...
İletişim (e-posta)	: ibrahim.halil.demir91@gmail.com
Yayınları	:

Demir. İ. H., Kaya. N. 2018. Araç bagaj hacminin hesaplanması için bir algoritma geliştirilmesi. 9 th International Automotive Technologies Congress, OTEKON 2018, 07-08 Mayıs 2018, Bursa.