Matematiksel Modelleme, Benzetişim, Denetim Kuramları ve Uygulamaları Dergisi
(The International Journal of Mathematical Modeling, Simulation, Control Theories and Applications)
ISSN *(International Standard Serial Number)*: 2602-4691
Cilt *(Volume)*: 1, Sayı *(Issue)*: 1, Sayfalar *(Pages)*: 1-8, Yıl *(Year)*: 2026
Dijital Nesne Tanımlayıcı *(Digital Object Identifier)*: https://doi.org/10.67489/mbh4691.2026.1

BİRİNCİ DERECE TEK-GİRDİLİ TEK-ÇIKTILI (TGTÇ) BİR PROSESİN SÜREKLİ ZAMANLI VE AYRIK ZAMANLI SİMÜLASYONLARININ KARŞILAŞTIRILMASI

(Comparison of Continuous-Time and Discrete-Time Simulations of a Single-Input Single-Output (SISO) First-Order Process)

Ercüment Özer *

* Dr. Öğr. Üy., Biyomedikal Mühendisliği Bölümü, Mühendislik Mimarlık Fakültesi,
İstanbul Yeni Yüzyıl Üniversitesi, İstanbul, Türkiye. ercument.ozer@yeniyuzyil.edu.tr, drozer@protonmail.com
(Asst. Prof. Dr., Biomedical Engineering Department, College of Engineering and Architecture, Istanbul Yeni Yuzyil University, Istanbul, Turkey)

Alınma Tarihi: 1 Haziran 2026 Kabul Tarihi: 15 Haziran 2026 Yayın Tarihi: 3 Temmuz 2026
(Received: 1 June 2026 Accepted: 15 June 2026 Published: 3 July 2026)

Araştırma Makalesi
(Research Paper)

ÖZET

Bu çalışmada, 1'nci derece bir sıradan diferansiyel denklem ile ifade edilebilen Tek-Girdili Tek-Çıktılı (TGTÇ) bir prosesin sürekli zamanlı ve ayrık zamanlı simülasyon performansları karşılaştırıldı. Sürekli zamanlı simülasyonlar için 1'nci Derece Euler İntegrasyon Yöntemi (Euler) ve 4'üncü Derece Runge-Kutta (Runge-Kutta) Yöntemi kullanıldı. Kesikli zamanlı simülasyonlar için ise prosesin sıfır derece tutma (hold) eklenmiş kesikli (ayrık) transfer fonksiyonu kullanıldı. Testler, ekte yer alan C kodu kullanılarak gerçekleştirildi. Örneklemeler 1 zaman birimi örnekleme aralığı ile yapılırken, yöntemler 0,0000001'den 1'e kadar değişen farklı integrasyon ve kesikli model aralıklarıyla test edildi. Gerçekleştirilen sürekli zamanlı simülasyonlarda, Euler ve Runge-Kutta yöntemlerinin ufak integrasyon aralıkları için örtüşen ve isabetli sonuçlar ürettikleri fakat yüksek integrasyon aralıkları için, Euler yöntemi Runge-Kutta yönteminden daha isabetsiz olmak üzere, isabetsiz sonuçlar ürettikleri gözlemlendi. Hold eklenmiş ayrık transfer fonksiyonuyla yapılan simülasyonlar ise hem düşük, hem de yüksek aralıklar için isabetli sonuçlar verdi. Düşük integrasyon aralıklarında, Runge-Kutta simülasyonunun Euler simülasyonundan yaklaşık 2,7 kat daha uzun sürdüğü, ayrık transfer fonksiyonu ile gerçekleştirilen düşük test aralıklı simülasyonların ise Runge-Kutta yönteminden yaklaşık %23 daha uzun sürdüğü gözlendi. Ayrık transfer fonksiyonu kullanılarak yapılan kesikli zaman simülasyonları, her durumda ve kesikli model aralığı büyüklüğünden bağımsız olarak isabetli sonuçlar ürettiği için, örnekleme aralığının modelleme (simülasyon) aralığına eşit alındığı uzun örnekleme aralıklarıyla yapılan simülasyonlarda, hem Euler, hem de Runge-Kutta yöntemlerinden daha isabetli ve dolayısı ile de daha hızlı şekilde doğru sonuçlar üretmiştir.

Anahtar Kelimeler: *proses, simülasyon*

ABSTRACT

This study compares continuous-time and discrete-time simulations of a Single-Input Single-Output (SISO) process that can be expressed by a first-order ordinary differential equation. For continuous-time simulations, the First-Order Euler Integration Method (Euler) and the Fourth-Order Runge-Kutta Method (Runge-Kutta) were used. For discrete-time simulations, the discrete transfer function of the process with a zero-order hold was used. The tests were performed using the C code provided in the appendix. Sampling was done at intervals of 1 time units, while the methods were tested at different integration and discrete model intervals ranging from 0.0000001 to 1. In continuous-time simulations, it is observed that the Euler and Runge-Kutta methods produced overlapping and accurate results for short integration intervals, but for long integration intervals, both methods produced inaccurate results, with the Runge-Kutta being less inaccurate. Simulations performed with the discrete transfer function that includes a hold yielded accurate results for both short and long intervals. It is observed that for low integration intervals, the Runge-Kutta method took approximately 2.7 times longer than the Euler method, and simulations with discrete transfer function at low test intervals took approximately 23% longer than the Runge-Kutta method. However, discrete-time simulations using discrete transfer function produced accurate values in all cases regardless of the length of the modeling intervals. Therefore, in simulations with long sampling intervals, where the sampling interval is taken as equal to the modeling (simulation) interval, it produced more accurate and thus faster correct results compared to both the Euler and Runge-Kutta methods.

Keywords: *process, simulation*

1. GİRİŞ

Dinamik simülasyon, çoğu zaman, çok değişkenli (iç etkileşimli), doğrusal olmayan, yüksek ölü zamanlı prosesler için denetleyici *(controller)* tasarımlarının entegre bir safhasıdır (Morari & Zafiriou, 1989; Luyben, 1996). Özellikle sadece tek bir ayar parametresi (filitre sabiti) bulunan dahili model denetleyicisi *(Internal Model Control)* tipindeki model öngörülü denetleyicilerin *(Model Predictive Controllers)* tasarımı için durum böyledir (Shin & Park, 1990; Arulalan & Deshpande, 1987; Garcia & Morari, 1985, Garcia & Morari 1982). Bu tür proseslerde, matematiksel proses modeli, kontrol edilen çıktının zamana karşı değişimini gösteren ve birbirlerine bağlı olan diferansiyel denklemler halinde ifade edilmektedir (Luyben, 1996; Seborg et al., 2004). Yüksek hassasiyet ve doğruluk istenildiği durumlarda, bu diferansiyel denklem-

lerin dinamik simülasyonları için genellikle 4'üncü Derece Runge-Kutta Yöntemi kullanılmaktadır (Luyben, 1996). Bazen de, simülasyon süresinin daha kısa olması avantajına sahip olan ama daha düşük hassasiyet ve doğrulukla çalıştığı bilinen 1'inci Derece Euler İntegrasyonu Yöntemi tercih edilmektedir (Luyben, 1996).

Bazı durumlarda, gerçek (fiziksel) olarak var olan proseslerin diferansiyel denklem formundaki matematiksel modelleri veya tasarım denklemleri mevcut değildir, sadece deneysel olarak elde edilebilen transfer fonksiyonları mevcuttur ve dinamik simülasyonların da bu transfer fonksiyonlarındaki "kazanç", "zaman sabiti", "ölü zaman" gibi parametreler kullanılarak yapılması gerekir (Holt & Morari, 1985; Stephanopoulos, 1984).

Bu çalışmada, sıfır derece tutma *(hold)* eklenmiş kesikli zaman transfer fonksiyonları *(Discrete Time Transfer Functions)* kullanılarak yapılabilecek dinamik simülasyonların 4'üncü Derece Runge-Kutta ve 1'inci Derece Euler yöntemleri ile aynı derecede hassas ama onlara daha hızlı bir alternatif olup olamayacağı 1'nci derece Tek-Girdili Tek-Çıktılı (TGTÇ) basit bir stabil proses üzerinde karşılaştırmalı olarak incelenmiştir. Bu çalışma için geliştirilmiş ve kullanılmış bulunan C yazılımı kodu da bu makalenin ekinde beyan edilmiştir.

Bu çalışmanın devam makalelerinde, aynı yaklaşımın, transfer fonksiyonları bilinen ölü zamanlı ve çok değişkenli prosesler için de etkili şekilde kullanılıp kullanılamayacağı, kapalı ve açık çevrim kontrol döngülerinin dinamik simülasyonları ile birlikte karşılaştırmalı olarak incelenecektir.

2. MALZEME VE YÖNTEM

Bu çalışmada kullanılan temel araç ve yöntemler aşağıda başlıklar halinde açıklanmıştır.

4'ÜNCÜ DERECE RUNGE KUTTA YÖNTEMİ

Aşağıdaki kaynaklar bölümünde yer almakta olan "Luyben, 1996" kaynağında, hem 1'nci Derece Euler İntegrasyonu Yöntemi, hem de 4'üncü derece Runge-Kutta Yöntemi FORTRAN dilinde yazılmış kodlarla birlikte örnekli ve kapsamlı olarak açıklanmakta ve anlatılmaktadır.

1'nci Derece Euler yöntemindeki her bir integrasyon adımı, dt, 4'üncü Derece Ruge-Kutta yönteminde altı parçaya bölünmüştür. 1'nci adımda dt/6, ikinci adımda 2dt/6, üçüncü adımda 2dt/6 ve 4'üncü adımda dt/6 ilerleme gerçekleşir ve toplamda dt kadar ilerlenmiş olunur.

4'üncü Derece Runge Kutta döngüsündeki dört adım özetle aşağıda listelendiği gibidir. Aşağıdaki denklemlerde y_{eski}, y'nin (çıktının) Runge-Kutta hesaplamaları öncesindeki eski değeridir; y_{yeni} ise y'nin Runge-Kutta hesaplamaları sonrasındaki yeni değeridir; k_1, k_2, k_3, k_4 geçici değerleri ise her bir Runge-Kutta adımındaki y değişim miktarları ile ilgilidir.

4'üncü Derece Runge Kutta 1'nci Adım:

$$(dy/dt)_1 = f(y_{eski})$$
$$k_1 = (dy/dt)_1\ (dt/6)$$
$$y_1 = k_1 + y_{eski}$$

4'üncü Derece Runge Kutta İkinci Adım:

$$(dy/dt)_2 = f(y_1)$$
$$k_2 = (dy/dt)_2\ (2\ dt)/6$$
$$y_2 = k_2 + y_1$$

4'üncü Derece Runge Kutta Üçüncü Adım:

$$(dy/dt)_3 = f(y_2)$$
$$k_3 = (dy/dt)_3\ (2\ dt)/6$$
$$y_3 = k_3 + y_2$$

4'üncü Derece Runge Kutta 4'üncü Adım:

$$(dy/dt)_4 = f(y_3)$$
$$k_4 = (dy/dt)_4\ (dt/6)$$
$$y_{yeni} = k_4 + y_3$$

Bu çalışmada, aşağıda açıklandığı ve bulgularla gösterildiği üzere, dt değeri yeterince küçük seçildiğinde, 4'üncü Derece Runge-Kutta ile elde edilen değerler ile 1'inci Derece Euler ile elde edilen değerler arasında belirgin bir fark gözlenmemiştir fakat artan hesaplama adımları nedeniyle 4'üncü Derece Runge-Kutta simülasyonu göreceli olarak daha yavaş gerçekleşmiştir.

Bu çalışmadaki dinamik simülasyonlar C programlama dili kullanılarak hem 1'nci Derece Euler İntegrasyonu, hem de 4'üncü derece Runge-Kutta İntegrasyonu yöntemleriyle karşılaştırmalı olarak yapılmıştır. Bu çalışma için yazılmış olan ekteki kod, Microsoft Windows 10 Pro işletim sistemi ve Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, 2501 Mhz, 4 Çekirdek, 4 Mantıksal işlemci üzerinde C++ derleyici kullanılarak çalıştırılmıştır.

SİMULASYON DEĞİŞKENLERİ ve PARAMETRELERİ

Bu çalışma için şematik proses görüntüsü aşağıdaki şekildedir.

Şekil 1. Şematik proses görüntüsü;
girdi (etki) değişkeni m, çıktı (tepki) değişkeni y.

İntegrasyon (Türev) Aralığı:

Bu çalışmada standart integrasyon (türev) aralığı (dt) olarak 0,0000001 seçilmiştir (zaman biriminin on milyonda biri) fakat yazılımdaki bu değer istenildiğinde başka bir pozitif gerçek sayı ile değiştirilebilmektedir.

Örnekleme Aralığı:

Örnekleme aralığı değeri, yapılan dinamik simülasyondan hangi aralıklarla örnekleme yapılacağını gösteren değerdir. Örneğin, bu değer eğer 1 olarak seçilirse ve dt değeri de 0,0000001 ise, her 10 milyon simülasyon adımında (10 milyon integrasyon/türev aralığında) bir, o andaki proses çıktı değeri (y) kaydedilecek demektir.

Bu çalışmada, kesikli zaman modeli için belirlenen hesaplama aralığı T_{model} ile gösterilmektedir. Örneğin, eğer dt = 0,0000001 ise ve T_{model} = 1 ise, her 10 milyon sürekli zaman adımına karşılık kesikli zaman modeli sadece bir hesaplama yapacak demektir. Eğer, dt = T_{model} olarak seçilirse, sürekli ve kesikli modeller aynı sayıda yeni y değeri hesaplarlar.

Proses Kazancı (*Process Gain)*:

Aşağıda daha detaylı açıklandığı üzere, proses kazancı (***K*** veya $\boldsymbol{K_p}$), "a (dy/dt) + b y = c m" formundaki bir denklemle ifade edilen bir proses için, c/b = ***K*** değerine karşılık gelmektedir ve a, b, c değerleri girildiğinde yazılım tarafından otomatik olarak hesap edilmektedir. Bu çalışma için a=1, b=1, c=1 kullanılmıştır ama yazılımda farklı değerler de seçilebilmektedir.

Proses Zaman Sabiti *(Process Time Constant)*:

Aşağıda daha detaylı açıklandığı üzere, proses zaman sabiti ($\boldsymbol{\tau}$ veya $\boldsymbol{\tau_p}$), "a (dy/dt) + b y = c m" formundaki bir denklemle ifade edilen bir proses için, a/b = $\boldsymbol{\tau}$ değerine karşılık gelmektedir ve a, b, c değerleri verildiğinde yazılım tarafından otomatikman hesap edilmektedir. Bu çalışma için a=1, b=1, c=1 kullanılmıştır ama yazılımda farklı değerler de seçilebilmektedir.

DENKLEMLERIN DERİVASYONLARI

Tek-Girdili Tek-Çıktılı (TGTÇ) 1'nci Derece (Ölü Zamansız) Bir Prosesin Sürekli Zaman Bazındaki Temel Denklemleri

Birinci derece bir sıradan diferansiyel denklem *(Ordinary Differential Equation)* ile ifade edilebilen bir Tek-Girdili Tek-Çıktılı (TGTÇ) prosesin denklemi genel olarak aşağıdaki şekilde yazılabilir, (Ekhwan & Özer, 2026; Seborg et al., 2004; Luyben, 1996). Aşağıdaki kaynaklar bölümünde yer almakta olan "Stephanopoulos, 1984" kaynağında da, aşağıdaki derivasyonların detaylı ve farklı versiyonları bulunmaktadır. Aşağıdaki denklemdeki t "zaman", y "çıktı/tepki", m "girdi/etki" değişkenleridir.

$$a \frac{dy}{dt} + b\ y = c\ m$$

Yukarıdaki denklemdeki a, b ve c parametreleri b'ye bölündüğünde, τ = a/b ve K = c/b olmak üzere denklem aşağıdaki hale dönüşür.

$$\tau \frac{dy}{dt} + y = K\ m$$

Buradaki K kazanç, τ da zaman sabiti değerleridir. Yukarıdaki denklem dy/dt terimi eşitliğin bir tarafında yalnız bırakılacak şekilde yeniden düzenlendiğinde

$$\frac{dy}{dt} = \left(\frac{K}{\tau} m - \frac{1}{\tau} y\right) = f(m, y, K, \tau)$$

sonucu elde edilebilir. Denklem aşağıdaki şekilde yeniden düzenlenirse

$$\frac{dy}{dt} = \frac{\Delta y}{\Delta t} = \frac{y_{t+\Delta t} - y_t}{\Delta t} = f(m, y, K, \tau)$$

aşağıdaki iterasyon denklemi elde edilir:

$$y_{t+\Delta t} = f(m, y, K, \tau)\ \Delta t + y_t$$

Yukarıdaki eşitlik daha da açık şekilde yazılırsa aşağıdaki forma gelir ve bu son haliyle, başlangıç değerlerinden itibaren, direk olarak 1'inci Derece Euler İntegrasyon Yöntemi ile sürekli zamanlı dinamik simülasyon yapmakta kullanılabilir.

$$y_{t+\Delta t} = \left(\frac{K}{\tau} m_t - \frac{1}{\tau} y_t\right) \Delta t + y_t$$

Eşitlikteki Δt değeri, integrasyon aralığına (dt) karşılık gelmektedir ve sürekli simülasyonlarda daha isabetli sonuçlar için olabildiğince ufak bir pozitif sabit sayı olarak seçilmesi gerekir.

Kesikli Zaman Transfer Fonksiyonu Derivasyonu

Birinci derece bir sıradan diferansiyel denklem *(Ordinary Differential Equation)* ile ifade edilebilen Tek-Girdili Tek-Çıktılı (TGTÇ) ölü zamansız bir prosesin transfer fonksiyonu Laplace transformasyonu yoluyla aşağıdaki şekilde türetilebilmektedir. Bu derivasyonların detayları aşağıdaki kaynaklar bölümünde yer alan "Stephanopoulos, 1984" kaynağından temin edilebilir.

$$\tau \frac{dy}{dt} + y = K\ m$$

$$\tau\ L\left[\frac{dy}{dt}\right] + L\left[y\right] = K\ L\left[m\right]$$

$$\tau\ s\ L\left[y(t)\right] + L\left[y(t)\right] = K L\left[m(t)\right]$$

$$(\tau\ s + 1)\ \ L\left[y(t)\right] = K\ L\left[m(t)\right]$$

$$\frac{L\ [\ y(t)\ ]}{L\ [\ m(t)\ ]} = \frac{K}{\tau\ s + 1}$$

$$\overline{g}_{proses}(s) = \frac{\overline{y}(s)}{\overline{m}(s)} = \frac{K_{proses}}{\tau_{proses}\ s + 1}$$

Yukarıdaki y ve m değişkenlerinin sapma değişkenleri *(deviation variables)* olduklarının belirtilmesi için değişkenler $\boldsymbol{y}$ ve $\boldsymbol{m}$ üstü çizgili şekilde gösterilmişlerdir.

1'nci derece, ölü zamansız bir Tek-Girdili Tek-Çıktılı *(Single Input Single Output – SISO)* prosesin HOLD eklenmemiş ayrık transfer fonksiyonu aşağıdaki şekilde elde edilebilir. (Ekhwan & Özer, 2026; Seborg et al., 2004; Luyben, 1996; Stephanopoulos, 1984).

$$y(s) = \frac{K_p}{\tau_p\ s + 1}\ m(s)$$

$$\mathcal{Z}\left[\frac{1}{s + a}\right] = \frac{1}{1 - e^{-aT} z^{-1}}$$

olduğu için

$$\mathcal{Z}\left[\frac{K_p}{\tau_p\ s + 1}\right] = \mathcal{Z}\left[\frac{K_p/\tau_p}{s + 1/\tau_p}\right] = \frac{K_p/\tau_p}{1 - e^{-T/\tau_p} z^{-1}}$$

olur (T: örnekleme aralığı olmak üzere). Bu prosesin Sıfır Derece Tutma (HOLD) eklenmiş ayrık transfer fonksiyonu ise şöyle olacaktır,

$$\frac{y}{m} = Hg_p(z) = \frac{K_p\ (1 - e^{-T/\tau_p})\ z^{-1}}{1 - e^{-T/\tau_p}\ z^{-1}}$$

Bu durumda, kesikli zaman model denklemi aşağıdaki hale dönüşür. Bu çalışmada, T = dt olarak alındığında, aşağıdaki denklemin 1'inci Derece Euler ve 4'üncü Derece Runge-Kutta yöntemlerinin yerine kullanılabileceği gösterilmiştir.

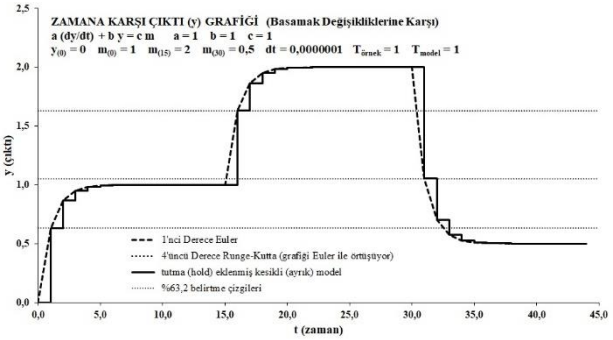
$$y = K_p\ (1 - e^{-T/\tau_p})\ m\ z^{-1} + e^{-T/\tau_p}\ y\ z^{-1}$$

3. BULGULAR

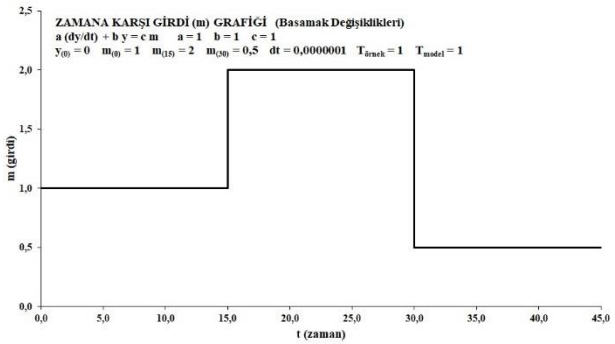
Bu makalede, amaca yönelik olarak, aşağıdaki grafiksel sonuçları veren örnek simülasyonlara yer verildi fakat ekte bulunan ve bu çalışma için yazılmış olan C kodu kullanılarak farklı simülasyonlar ve incelemeler de üretilebilir.

Bu çalışmada, ilk olarak, "a (dy/dt) + b y = c m" şeklindeki prosesin ilki 15'nci, ikincisi 30'uncu zaman birimlerinde verilen basamak değişikliklerine karşı verdiği tepki eğrileri 1'inci Derece Euler İntegrasyon Yöntemi, 4'üncü Derece Runge-Kutta Yöntemi ve kesikli (ayrık) zamanlı model simülasyonları üzerinden incelendi ve simülasyon süreleri karşılaştırıldı.

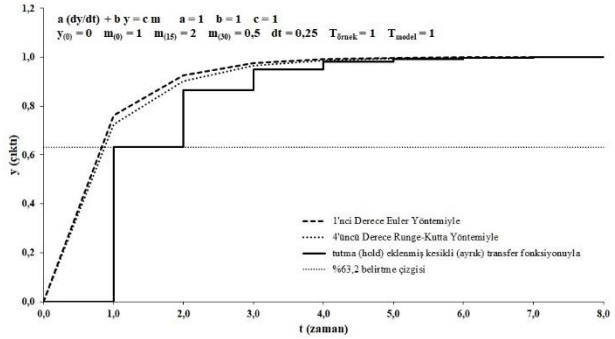
1'nci Derece Euler İntegrasyon Yöntemi kullanılarak dt=0,0000001, $T_{örnek}$=1 zaman birimi ile yapılan simülasyon 22,45 ± 5,61 saniye sürdü. Aynı simülasyon, 4'üncü Derece Runge-Kutta ile yapıldığında (dt = 0,0000001, $T_{örnek}$=1) 60,38 ± 3,29 saniye sürdü. Euler ve Runge-Kutta sonuçları bire bir örtüştü. Aynı simülasyon, kesikli (ayrık) zaman simülasyonu ile yapıldığında (dt = 0,0000001, T_{model} = 0,0000001, $T_{örnek}$ = 1 için) 67,25 ± 3,18 saniyede tamamlandı. Zaman değerleri dt = 0,0000001, T_{model} = 1, $T_{örnek}$ = 1 olarak verildiğinde ise için simülasyon 9,79 ± 0,64 saniye sürdü. Her iki simülasyonda da üretilmiş olan değerler hem birbirleri ile, hem de Euler ve Runge-Kutta sonuçları ile bire bir örtüştü.



Şekil 2.a: Zamana Karşı Çıktı (y) Grafiği. Kesikli model için simülasyon aralığı T_{model} = 1 olarak alınmışken, sürekli simülasyonlar için simülasyon aralığı dt = 0,0000001 olarak alınmıştır (örnekleme aralıkları = $T_{örnek}$ = 1). Sürekli zaman modeli için 1'inci Derece Euler ve 4'üncü Derece Runge-Kutta yöntemleri kullanılmıştır. Girdi (m) basamak değişimi grafiği Şekil 2.b grafiğindeki gibidir. 1'nci derece proseslerdeki basamak değişikliklerinde zaman sabitine ulaşma değerleri (% 63,2'ye ulaşılma değerleri) çizgilerle belirtilmiştir.

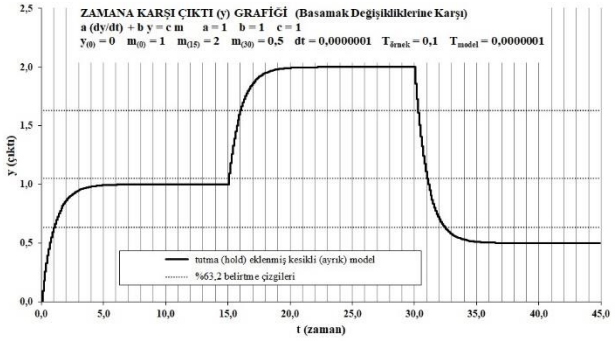


Şekil 2.b: Zamana Karşı Girdi (m) Grafiği. İki adet basamak değişikliği 15'nci ve 30'uncu zaman birimlerinde verilmiştir. Çıktı için başlangıç değeri $y_{(0)} = 0$, girdi için başlangıç değeri m = 1 alınmıştır. Diğer tüm simülasyon testlerinde de bu basamak değişiklikleri kullanılmıştır.



Şekil 3: Zamana Karşı Çıktı (y) Grafiği, Detay. Bu simülasyonda sürekli zaman simülasyon aralıkları dt = 0,25 zaman birimi olarak seçilmiştir. Tutma (hold) eklenmiş kesikli (ayrık) transfer fonksiyonu kullanılarak yapılan simülasyondaki simülasyon aralığı ise $T_{model} = 1$'dir. Runge-Kutta yöntemi Euler'e göre daha isabetli sonuçlar üretmiş olmakla birlikte, her iki yöntem de, dt = 0,25 için, transfer fonksiyonunun geniş simülasyon aralıklarına rağmen üretebilmiş olduğu isabetli sonuçları üretememişlerdir.

a (dy/dt) + b y = c m formundaki 1. derece bir diferansiyel denklemde, (c/b) = $\boldsymbol{K_p}$ = 1 ve (a/b) = $\boldsymbol{\tau_p}$ = 1 ise, m = 1 iken ve y'nin başlangıç değeri 0 iken, zaman = time = t = 1 oldugunda, $y = 1 - e^{-1}$ olmalıdır. Yani, t = 1 iken y = 0,632121 olmalıdır. Şekil 3'teki simülasyonda, transfer fonksiyonu tarafından üretilmiş bulunan ve zaman sabiti zamanına karşılık gelen 0,632121 değeri Runge-Kutta için 0,726297, Euler için ise 0,762695 olarak gerçekleşmiştir.



Şekil 4: Zamana Karşı Çıktı (y) Grafiği. Bu simülasyonda tutma (hold) eklenmiş kesikli (ayrık) transfer fonksiyonu kullanılarak yapılan simülasyondaki simülasyon aralığı T_{model} = 0,0000001'dir, örnekleme aralığı da 0,1'dir. Yani, kesikli simülasyon 0,0000001 aralıklarla yapılmış fakat 0,1 aralıklarla örneklendirilmiştir.

Yukarıdaki Şekil 3 ve 4, tutma eklenmiş transfer fonksiyonunun hem çok kısa model değeri hesaplama aralıkları için, hem de uzun aralıklar için tam isabetli sonuçlar üretebildiğini göstermektedir.

4. SONUÇ

Bu çalışmada, detayları ve gerekçeleri yukarıda gösterildiği üzere, ayrık transfer fonksiyonu kullanılarak yapılan kesikli zaman simülasyonları, her durumda ve kesikli model aralığı büyüklüğünden bağımsız olarak isabetli sonuçlar ürettiği için, örnekleme aralığının modelleme (simülasyon) aralığına eşit alındığı uzun örnekleme aralıklarıyla yapılan simülasyonlarda, hem Euler, hem de Runge-Kutta yöntemlerinden daha isabetli ve dolayısı ile de daha hızlı şekilde doğru sonuçlar ürettiği gözlemlenmiştir. Dolayısı ile, Tek-Girdili Tek-Çıktılı proseslerin simülasyonları için tutma (hold) eklenmiş ayrık (vurgulu) transfer fonksiyonlarının 1'nci Derece Euler İntegrasyon Yöntemi (Euler) ve 4'üncü Derece Runge-Kutta (Runge-Kutta) Yöntemlerine alternatif olarak kullanılabileceği gözlemlenmiştir.

5. KAYNAKLAR

Arulalan, R. & Deshpande, P.B. (1987). "Simplified Model Predictive Control", Ind. Eng. Chem. Res, American Chemical Society, 26(2):347-356.

Garcia, C.E. & Morari, M. (1982). "Internal Model Control. 1. A Unifying Review and Some New Results", Ind. Eng. Chem. Process Des. Dev., American Chemical Society, 21(2):308-323.

Garcia, C.E. & Morari, M. (1985). "Internal Model Control. 2. Design Procedure for Multivariable Systems", Ind. Eng. Chem. Process Des. Dev., American Chemical Society, 24:472-484.

Garcia, C.E. & Morari, M. (1985). "Internal Model Control. 3. Multivariable Control Law Computation and Tuning Guidelines", Ind. Eng. Chem. Process Des. Dev., American Chemical Society, 24:484-494.

Holt, B.R. & Morari, M. (1985) "Design of Resilient Processing Plants-V The Effect of Deadtime on Dynamic Resilience", Chemical Engineering Science, Pergamon Press Ltd., 40(7):1229-1237.

Luyben, W.L. (1996). Process Modeling, Simulation and Control for Chemical Engineers - 2nd Ed., McGraw-Hill Book Co., New York, USA.

Morari, M. & Zafiriou, E. (1989). Robust Process Control, Prentice-Hall Inc., New York, USA.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A. (2004). Process Dynamics and Control - 2nd Ed., John Wiley & Sons Inc., New York, USA.

Shin, S.S. & Park, S.W. (1990). "IMC Controller Design for Multivariable Systems", Korean J. of Chem. Eng., 7(2) 115-125.

Stephanopoulos, G. (1984). Chemical Process Control - An Introduction to Theory and Practice, Prentice Hall, New Jersey, USA.

Şenen Al Ekhwan, R. & Özer, E. (2026). "Dahili Model Kontrol Tabanlı ve Oransal İntegral Türevsel Tabanlı Denetleyici Tasarımlarının Yapay Pankreas Kontrolü İçin Karşılaştırılması", *İYYÜ Mühendislik Mimarlık Fakültesi Dergisi, Hakemli-Akademik, 1*(1):33-48. https://doi.org/10.66125/0.2026.6

Bu makaleye atıfta bulunmak için:

Özer, E. (2026). "Birinci Derece Tek-Girdili Tek-Çıktılı (TGTÇ) Bir Prosesin Sürekli Zamanlı ve Ayrık Zamanlı Simülasyonlarının Karşılaştırılması", *Matematiksel Modelleme, Benzetişim, Denetim Kuramları ve Uygulamaları Dergisi, Hakemli-Akademik*, *1*(1):1-8. https://doi.org/10.67489/mbh4691.2026.1

6. EKLER

```
/*  BU PROGRAM, C DILINDE, a (dy/dt) + b y = c m FORMUNDAKI BIRINCI DERECE BIR TGTC (TEK-GIRDILI TEK-CIKTILI) PROSESIN
    4ncu DERECE RUNGE-KUTTA YONTEMI ILE ve 1nci DERECE EULER INTEGRASYONU YONTEMI ILE SUREKLI ZAMAN DINAMIK SIMULASYONLARINI,
    PROSESIN SIFIR DERECE TUTMA (HOLD) EKLENMIS AYRIK TRANSFER FONKSIYONU UZERINDEN DE KESIKLI ZAMAN SIMULASYONUNU YAPAR
    DR. OGR. UYESI ERCUMENT OZER  */

/*  KUTUPHANELERIN EKLENMESI  */
    #include <stdio.h>   /*  C standart kutuphanesi  */
    #include <math.h>    /*  matematik kutuphanesi  */
    #include <time.h>    /*  zaman kutuphanesi  */

/*  TUREV FONKSIYONUNUN TANIMLANMASI, dy/dt = turev = f(m,y)  */
    long double fturev(long double m, long double y, long double a, long double b, long double c) {
     long double turev = 0.0  ;
                 turev = (c/a)*m - (b/a)*y  ;
          return turev  ;  }

/*  PROGRAM BASLANGICI  */
    int main() {

/*  DEGISKENLERIN TANIMLANMASI  */
    long double m   ;   /*  a (dy/dt) + b y = c m  denklemindeki  input/girdi/etki  degiskeni  */
    long double y   ;   /*  a (dy/dt) + b y = c m  denklemindeki output/cikti/tepki degiskeni  */

    long double a, b, c  ;   /*  a (dy/dt) + b y = c m  denklemindeki parametreler  */

    long double dydt1, dydt2, dydt3, dydt4         ; /* turevlerin sayisal degerlerini gecici olarak saklamak icin degiskenler */
    long double k1, k2, k3, k4, geciciy, turev=0.0 ; /* 4. Derece Runge-Kutta uygulamasi icin gereken diger degiskenler */
    long double yeski                              ; /* toplam turev hesaplamak icin eski y degiskeni degerini saklama degiskeni */

    int secim                  ;   /*  VARSAYILAN degerlerin secilip secilmemesi tercihi ile ilgili degisken  */
    int yontem=1               ;   /*  yontem secimi ile ilgili degisken  */
                                   /*  yontem=1 ise EULER (varsayilan)  */
                                   /*  yontem=2 ise RUNGE-KUTTA  */
                                   /*  yonetm=3 HOLD EKLENMIS AYRIK MODEL (TRANSFER FONKSIYONU)  */

    long double t           ;   /*  zaman (time)  */
    long double tbaslangic  ;   /*  simulasyonun baslatilma zamani  */
    long double tson        ;   /*  simulasyonun bitirilme zamani  */

    long double dt          ;   /*  zaman artisi miktari ve/veya integrasyon araligi  */

    long double     ModelZamaniBaslangici,     ModelZamani,     ModelAraligi, Tmodel  ;
    long double OrneklemeZamaniBaslangici, OrneklemeZamani, OrneklemeAraligi, Tornek  ;
    long double   KontrolZamaniBaslangici,   KontrolZamani,   KontrolAraligi, T       ;

    long double Kp              ;   /*  proses kazanci (gain)  */
    long double Taup            ;   /*  proses zaman sabiti (time constant)  */

    long double yz, yz1, mz1    ;   /*  kesikli zamanli fonksiyon degiskenleri  */
                                   /*  yz1 ve mz1 bir onceki y ve bir onceki m degerleri anlamindadir  */

    long double monceki, yzonceki, turevonceki  ;  /* bazi eski degerlerin saklanmasi icin degiskenler */
    long double t0, m0, y0, yz0, turev0, turevz  ;   /*  ilk degerler  */

/*  SONUCLARIN YEDEKLENECEGI DOSYANIN ACTIRILMASI
    burada, C:\\Users\\Moore\\Desktop\\makale\\  yerine,
    simulasyonun yapilacagi bilgisayardaki bir klasorun adresi yazilmalidir,
    gerekirse burada oldugu gibi "\" YERINE "\\" KULLANILMALIDIR  */
    FILE *ciktilar  ;
    ciktilar = fopen("C:\\Users\\Moore\\Desktop\\makale\\ciktilar-simulasyon-SISO.txt","w") ;

/*  SIMULASYON/INTEGRASYON ARALIGININ TANIMLANMASI  */
    dt = 0.0000001  ;   /*  zaman artisi miktari ve/veya integrasyon araligi  */

/*  ORNEKLEME ARALIGININ TANIMLANMASI
    OrneklemeAraligi degeri dt degerinden BUYUK veya ona ESIT bir pozitif gercek sayi olmalidir
    ve tercihen OrneklemeAraligi degeri dt degerinin tam sayi bir kati olmalidir  */

    OrneklemeZamaniBaslangici = 0.0                       ;
             OrneklemeZamani = OrneklemeZamaniBaslangici  ;
            OrneklemeAraligi = 0.1                        ;

                      Tornek = OrneklemeAraligi           ;

/*  MODEL TEPKILERI ARALIGININ TANIMLANMASI
    ModelAraligi degeri OrneklemeAraligi degerinden BUYUK veya ona ESIT bir pozitif gercek sayi olmalidir
    ve tercihen ModelAraligi degeri OrneklemeAraligi degerinin tam sayi bir kati olmalidir.  */

        ModelZamaniBaslangici = 0.0                       ;
                 ModelZamani = ModelZamaniBaslangici      ;
                ModelAraligi = 0.0000001                        ;

                      Tmodel = ModelAraligi               ;

/*  KONTROL ARALIGININ TANIMLANMASI  */
      KontrolZamaniBaslangici = 0.0                       ;
               KontrolZamani = KontrolZamaniBaslangici    ;
              KontrolAraligi = 1.0                        ;

                           T = KontrolAraligi             ;

/*  VARSAYILAN BASLANGIC DEGERLERI  */
            m =  1.0        ;   /*  denklemdeki input/girdi/etki degiskeni varsayilan ilk degeri  */
            y =  0.0        ;   /*  denklemdeki output/cikti/tepki degiskeni varsayilan ilk degeri  */

/*  ZAMAN ILE ILGILI BASLANGIC DEGERLERI  */
    tbaslangic =  0.0       ;   /*  zaman 0'dan baslatiliyor ama baska pozitif bir deger de secilebilir  */
          tson = 45.0       ;   /*  simulasyonun bitirilecegi zaman  */
             t = tbaslangic ;   /*  zaman ilk degeri  */
```

```
/*  a (dy/dt) + b y = c m  PARAMETRE DEGERLERI  */
    a = 1.0   ;
    b = 1.0   ;
    c = 1.0   ;

/*  KESIKLI ZAMAN MODELI BASLANGIC DEGERLERI  */
    Taup = a/b  ;
      Kp = c/b  ;
     yz  = y    ;
     yz1 = yz   ;
     mz1 = m    ;

/*  VARSAYILAN DEGERLERIN EKRANDAN DEGISTIRILMESI  */
    printf("\nVARSAYILAN DEGERLERIN EKRANDAN DEGISTIRILMESI:\n\n") ;
    printf("VARSAYILAN degerleri kabul etmek icin 1, kendi degerlerinizi ekrandan girmek icin 0 degerini veriniz = ? ") ; scanf("%d",
&secim) ;
    if (secim==1) { goto DEVAM ; } ;
    printf("\n");

    printf("YONTEM SECIMI (yontem=1 EULER , yontem=2 RUNGE-KUTTA, yontem=3 MODEL) = ? ") ; scanf("%d", &yontem) ;
    printf("\n");

    printf("m (m degiskeninin degerini giriniz) (VARSAYILAN: 1.0) = ? ") ; scanf("%Lf", &m) ;
    printf("y (y degiskeninin baslangic degerini giriniz) (VARSAYILAN: 0.0) = ? ") ; scanf("%Lf", &y) ;
       yz  = y    ;
       yz1 = yz   ;
       mz1 = m    ;
    printf("\n");

    printf("a (a degiskeninin baslangic degerini giriniz) (VARSAYILAN: 1.0) = ? ") ; scanf("%Lf", &a) ;
    printf("b (b degiskeninin baslangic degerini giriniz) (VARSAYILAN: 1.0) = ? ") ; scanf("%Lf", &b) ;
    printf("c (c degiskeninin baslangic degerini giriniz) (VARSAYILAN: 1.0) = ? ") ; scanf("%Lf", &c) ;
      Taup = a/b  ;
       Kp = c/b  ;
    printf("\n");

    printf("tbaslangic (simulasyonun baslangic zamanini giriniz) (pozitif bir deger olmali) (VARSAYILAN:  0.0) = ? ") ; scanf("%Lf",
&tbaslangic   ) ;
    t = tbaslangic  ;
    printf("tson       (simulasyonun bitis     zamanini giriniz) (tbaslangic'tan buyuk bir pozitif deger olmali) (VARSAYILAN: 45.0) = ?
") ; scanf("%Lf", &tson) ;
    printf("\n");

    printf("dt (zaman artisi miktari ve/veya integrasyon araligi, cok ufak bir pozitif gercek sayi olmalidir) (VARSAYILAN: 0.0000001) =
? ") ; scanf("%Lf", &dt) ;
    printf("ModelAraligi (model degerlerinin hesaplanma araligini giriniz) (VARSAYILAN: 1.0) = ? ") ; scanf("%Lf", &ModelAraligi) ;
           Tmodel = ModelAraligi  ;
    printf("OrneklemeAraligi (ornekleme araligini giriniz) (integrasyon araligindan buyuk bir pozitif sayi olmali) (VARSAYILAN: 1.0) =
? ") ; scanf("%Lf", &OrneklemeAraligi) ;
           Tornek = OrneklemeAraligi  ;
    printf("KontrolAraligi (kontrol araligini giriniz) (integrasyon araligindan buyuk bir pozitif sayi olmali) (VARSAYILAN: 1.0) = ? ")
; scanf("%Lf", &KontrolAraligi) ;
              T = KontrolAraligi  ;
    printf("\n");

/*  VARSAYILAN DEGERLERLE DEVAM ETME TERCIHI ICIN DEVAM ADINDA BIR DONGU BASLANGICI ADI TANIMLANDI  */
    DEVAM:
    printf("\n");

/*  BAZI ILK DEGERLERIN TANIMLANMASI   */
       monceki = m      ;
      yzonceki = yz     ;
    turevonceki = turev  ;

/*  ILK DEGERLER  */
       t0 = t      ;
       m0 = m      ;
       y0 = y      ;
      yz0 = yz     ;
    turev0 = turev  ;
    turevz = turev  ;

/*  PERFORMANS OLCMEK ICIN BASLANGIC ZAMAN/SAAT KAYITI  */
    clock_t baslangic = clock();

/*  ILK DEGERLERIN EKRANA ve DOSYAYA YAZDIRILMASI  */
    if (yontem==1) {
 printf(          "t= %10.4Lf  m= %6.2Lf  y (EULER)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, y0, turev0)       ;
fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (EULER)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, y0, turev0)       ; } ;
    if (yontem==2) {
 printf(          "t= %10.4Lf  m= %6.2Lf  y (RUNGE-KUTTA)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, y0, turev0) ;
fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (RUNGE-KUTTA)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, y0, turev0) ; } ;
    if (yontem==3) {
 printf(          "t= %10.4Lf  m= %6.2Lf  y (MODEL)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, yz0, turev0)      ;
fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (MODEL)= %9.6Lf  Turev= %8.5Lf \n", t0, m0, yz0, turev0)      ; } ;

/*  SONRAKI ZAMANLARIN HESAPLANMASI  */
       ModelZamani = tbaslangic +     ModelZamani + Tmodel  ;  /* SONRAKI MODEL TEPKISI ZAMANININ HESAPLANMASI */
    OrneklemeZamani = tbaslangic + OrneklemeZamani + Tornek  ;  /* SONRAKI ORNEKLEME ZAMANININ HESAPLANMASI */
      KontrolZamani = tbaslangic +   KontrolZamani + T       ;  /* SONRAKI KONTROL ZAMANININ HESAPLANMASI */

/*  SIMULASYON DONGUSUNUN BASLANGICI, BU DONGUYE dongu ADI VERILDI  */
    dongu:

/*  y NIN ESKI DEGERININ SAKLANMASI (TOPLAM TUREVI HESAPLAMAK ICIN)  */
     yeski = y  ;

/*  ********************************************************************************  */
/*  ******************  1nci DERECE EULER SIMULASYONU BASLANGICI   ******************  */

/*  BIR SONRAKI y DEGERINI EULER YONTEMI KULLANARAK HESAPLAMAK ICIN  */
    if (yontem==1) { y = fturev(m,y,a,b,c)*dt + y  ;
```

```
/*  ********************************************************************  */
/*  *********************    YAZDIRMA BASLANGICI    *********************  */

     if (t>=(OrneklemeZamani-0.00000001)) {

/*  DEGERLERIN DOSYAYA ve EKRANA YAZDIRILMASI  */
    fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (EULER)= %9.6Lf  Turev= %8.5Lf \n", t, m, y, turev) ;
     printf(          "t= %10.4Lf  m= %6.2Lf  y (EULER)= %9.6Lf  Turev= %8.5Lf \n", t, m, y, turev) ;

/*  BIR SONRAKI ORNEKLEME ANININ (ZAMANININ) HESAPLANMASI  */
    OrneklemeZamani = OrneklemeZamani + Tornek  ;  }

/*  ************************    YAZDIRMA SONU    ************************  */
/*  ********************************************************************  */

    goto EULER  ;  }  ;

/*  ********************     1nci DERECE EULER SIMULASYONU SONU      ********************  */
/*  ***************************************************************************************  */

/*  ***************************************************************************************  */
/*  ***************** 4uncu DERECE RUNGE- KUTTA SIMULASYONU BASLANGICI ******************  */

/*  BIR SONRAKI y DEGERINI 4. DERECE RUNGE-KUTTA KULLANARAK HESAPLAMAK ICIN  */
    if (yontem==2) {

     dydt1 = fturev(m,y,a,b,c)              ;    /*  ilk turev parcasi  */
        k1 = dydt1*(dt/6.0)                 ;    /*  ilk agirlikli degisim  */
   geciciy = k1 + y                         ;    /*  gecici y  */

     dydt2 = fturev(m,geciciy,a,b,c)        ;    /*  ikinci turev parcasi  */
        k2 = dydt2*(2.0*dt/6.0)             ;    /*  ikinci agirlikli degisim  */
   geciciy = k2 + geciciy                   ;    /*  gecici y  */

    dydt3 = fturev(m,geciciy,a,b,c)         ;    /*  ucuncu turev parcasi  */
       k3 = dydt3*(2.0*dt/6.0)              ;    /*  ucuncu agirlikli degisim  */
  geciciy = k3 + geciciy                    ;    /*  gecici y  */

    dydt4 = fturev(m,geciciy,a,b,c)         ;    /*  dorduncu turev parcasi  */
       k4 = dydt4*(dt/6.0)                  ;    /*  dorduncu agirlikli degisim  */
        y = k4 + geciciy                    ;    /*  y  */

/*  ********************************************************************  */
/*  *********************    YAZDIRMA BASLANGICI    *********************  */

      if (t>=(OrneklemeZamani-0.00000001)) {

/*  DEGERLERIN DOSYAYA ve EKRANA YAZDIRILMASI  */
    fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (RUNGE-KUTTA)= %9.6Lf  Turev= %8.5Lf \n", t, m, y, turev) ;
     printf(          "t= %10.4Lf  m= %6.2Lf  y (RUNGE-KUTTA)= %9.6Lf  Turev= %8.5Lf \n", t, m, y, turev) ;

/*  BIR SONRAKI ORNEKLEME ANININ (ZAMANININ) HESAPLANMASI  */
    OrneklemeZamani = OrneklemeZamani + Tornek  ;  }

/*  ************************    YAZDIRMA SONU    ************************  */
/*  ********************************************************************  */

    goto RUNGEKUTTA  ;  }  ;

/*  *****************    4uncu DERECE RUNGE- KUTTA SIMULASYONU SONU    *****************  */
/*  ***************************************************************************************  */

/*  ***************************************************************************************  */
/*  ******************** KESIKLI ZAMAN MODELI SIMULASYONU BASLANGICI ********************  */

/*  BIR SONRAKI y DEGERINI SADECE KESIKLI ZAMAN MODELI KULLANARAK HESAPLAMAK ICIN  */

      if (yontem==3) { if (t>=(ModelZamani          )) {

/*  KESIKLI FONKSIYON DEGERININ HESAPLANMASI ICIN
    SIFIR DERECE TUTMA (HOLD) EKLENMIS KESIKLI TRANSFER FONKSIYONUNUN KULLANILMASI  */
         yz = Kp*(1.0-expl(-Tmodel/Taup)) * mz1 + expl(-Tmodel/Taup) * yz1   ;
        yz1 = yz  ;
        mz1 = m   ;

   turevz = (yz-yzonceki) / Tornek ;   /*  MODEL icin net turev  */

/*  ********************************************************************  */
/*  *********************    YAZDIRMA BASLANGICI    *********************  */

      if (t>=(OrneklemeZamani          )) {

/*  "ESKI KESIKLI FONKSIYON DEGERI ILE KESIKLI FONKSIYON DEGERININ ve GERCEK DEGERIN DOSYAYA YAZDIRILMASI  */
    fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (MODEL)= %9.6Lf  Turev= %8.5Lf \n", t, monceki, yzonceki, turevonceki) ;

/*  "YENI KESIKLI FONKSIYON DEGERI ILE" KESIKLI FONKSIYON DEGERININ ve GERCEK DEGERIN DOSYAYA YAZDIRILMASI */
    fprintf(ciktilar, "t= %10.4Lf  m= %6.2Lf  y (MODEL)= %9.6Lf  Turev= %8.5Lf \n", t, m, yz, turevz) ;

/*  "YENI KESIKLI FONKSIYON DEGERI ILE" KESIKLI FONKSIYON DEGERININ ve GERCEK DEGERIN EKRANA YAZDIRILMASI */
     printf(          "t= %10.4Lf  m= %6.2Lf  y (MODEL)= %9.6Lf  Turev= %8.5Lf \n", t, m, yz, turevz) ;

        monceki = m       ;
       yzonceki = yz      ;
    turevonceki = turevz  ;

/*  BIR SONRAKI ORNEKLEME ANININ (ZAMANININ) HESAPLANMASI  */
    OrneklemeZamani = OrneklemeZamani + Tornek  ;  }

/*  ************************    YAZDIRMA SONU    ************************  */
/*  ********************************************************************  */

//        monceki = m       ;
```

```
//       yzonceki = yz      ;
//    turevonceki = turevz  ;

/*  BIR SONRAKI MODEL DEGERI HESAPLAMA ANININ (ZAMANININ) HESAPLANMASI  */
    ModelZamani = ModelZamani + Tmodel  ;  }

    goto MODEL  ;  }  ;

/*  ********************    KESIKLI ZAMAN MODELI SIMULASYONU SONU    ********************  */
/*  ***************************************************************************************  */

EULER:
RUNGEKUTTA:
MODEL:

    turev  = (y -yeski   ) / dt     ;   /*  EULER ve RUNGE-KUTTA icin net turev  */

/*  t'NIN DEGERININ dt KADAR ARTTIRILMASI (YENI ZAMAN DEGERININ HESAPLANMASI)  */
    t = t + dt  ;

/*  BASAMAK DEGISIKLIKLERININ VERILMESI (EULER ve RUNGE-KUTTA SIMULASYONU ICIN)
    basamak degisiklikleri zamanlarini OrneklemeAraligi zamaninin pozitif tam sayi bir kati olarak seciniz  */
    if ((yontem==1)||(yontem==2)) {
    /* m = m * expl(-0.69314718*dt/Tyarilanma) ; yarilanma zamani varsa verilmesi, logl(2) = 0.69314718 */
    if ( t>= (15.0+tbaslangic           ) ) {m=2.0;}  ;
    if ( t>= (30.0+tbaslangic-0.00000001) ) {m=0.5;}  ;  }  ;

/*  BASAMAK DEGISIKLIKLERININ VERILMESI (KESIKLI ZAMAN MODELI SIMULASYONU ICIN)
    basamak degisiklikleri zamanlarini ModelAraligi zamaninin pozitif tam sayi bir kati olarak seciniz  */
    if (yontem==3) {
    if ( t>= (15.0+tbaslangic           ) ) {m=2.0;}  ;
    if ( t>= (30.0+tbaslangic-0.00000001) ) {m=0.5;}  ;  }  ;

/*  SIMULASYONU SONLANDIRMA SATIRI  */
    if (t>tson) { goto bitir  ; }  ;

/*  SIMULASYON DONGUSUNUN BASINA GERI GITMEK ICIN  */
    goto dongu;

/*  PROGRAMI SONLANDIRMAK ICIN  */
    bitir:

/*  PERFORMANS HESAPLAMAK ICIN BITIS ZAMAN/SAAT KAYITI  */
    clock_t bitis = clock();

/*  PERFORMANS SURE HESAPLAMASI  */
    long double cpu_time_used = ((long double) (bitis - baslangic)) / CLOCKS_PER_SEC  ;

/*  PERFORMANS SURESININ EKRANA ve DOSYAYA YAZDIRILMASI */
     printf(          "\nExecution Time: %10.5Lf saniye\n", cpu_time_used)  ;
    fprintf(ciktilar, "\nExecution Time: %10.5Lf saniye\n", cpu_time_used)  ;

/*  SONUCLARI YAZDIRMAK ICIN ACTIRILAN DOSYANIN KAPATILMASI  */
    fclose(ciktilar)  ;

/*  PROGRAM SONU  */
    return 0 ;  }
```