APLIKASI SISTEM ODE DALAM POPULATION GROWTH

Joshua Evans Setiyawan[1], *Lie Reubensto*[2], *Roger Julianto Angryawan*[3]

[1,2,3] School of Computer Science (SoCS) Universitas Bina Nusantara Kampus Anggrek, Jl. Raya Kb. Jeruk No.27, RT.1/RW.9, Kemanggisan, Kec. Palmerah, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta 11530

NIM: 2501972354[1], 2540124633[2], 2501962990[3]

Ⅰ. Pendahuluan

Persamaan diferensial secara sederhana merupakan persamaan yang mengandung fungsi tak diketahui dengan turunannya. Atau secara formal, persamaan diferensial merupakan "persamaan matematika untuk fungsi satu atau lebih variabel yang tidak diketahui yang menghubungkan nilai-nilai fungsi itu sendiri dan turunannya dari berbagai orde. Persamaan diferensial sendiri terbagi menjadi dua kategori, yaitu persamaan diferensial biasa dan persamaan diferensial parsial di mana persamaan diferensial biasa akan berfokus pada satu variabel independen sedangkan persamaan diferensial parsial akan berfokus pada lebih dari satu variabel independen. Persamaan diferensial yang hanya memiliki turunan pertama disebut sebagai persamaan diferensial orde pertama.

Persamaan diferensial memiliki banyak pengaplikasian yang dapat digunakan dalam kehidupan sehari-hari, baik dari bidang *engineering*, fisika, biologi, ekonomi, dan lain sebagainya. Begitu juga halnya untuk persamaan diferensial orde pertama. Karthikeyan dan Shrinivasan mempelajari bahwa persamaan diferensial orde pertama memiliki banyak aplikasi untuk perpindahan area panas pada benda padat. Begitu juga dengan Hassan A. dan Zakari Y. yang mempelajari bahwa persamaan diferensial orde pertama memiliki banyak aplikasi pada masalah temperatur.

Pada artikel ilmiah ini, kami akan menelusuri masalah *Population Growth* ke dalam bentuk model matematika dengan memanfaatkan sistem persamaan diferensial orde pertama. Kami juga akan memberikan solusi analitik dan numerik untuk mendapatkan persamaan umum dari model yang telah kami buat. Untuk solusi numeriknya, kami akan menggunakan bahasa pemrograman Python dan mengerjakannya dengan metode Runge-Kutta.

Ⅱ. Pemodelan Matematika

Untuk merepresentasikan masalah *Population Growth* ke dalam bentuk model matematika, ada beberapa faktor yang perlu untuk diketahui terlebih dahulu. Ketika berbicara tentang pertumbuhan, pasti ada sumber pendukung yang mendorong pertumbuhan tersebut. Asumsi, kita sebut sumber pendukung sebagai "Sumber Daya Alam (SDA)" atau "Resource (R)". Jika SDA meningkat, jumlah populasi (P) juga akan meningkat. Sedangkan, dengan meningkatnya jumlah populasi, jumlah SDA akan menurun. Dengan menurunnya jumlah SDA, jumlah populasi juga akan menurun. Ketika jumlah populasi menurun, jumlah SDA bisa bertambah dan siklus terus-menerus berlanjut. Ini menjadi faktor-faktor utama yang harus diperhatikan.

Sehubungan dengan hal tersebut, ada juga faktor-faktor lain. Jumlah SDA dan populasi diasumsi memiliki tingkat kenaikan/penurunan dan tidak terjadi secara instan. Kami akan sebut faktor-faktor ini sebagai “Resource Growth Rate (RGR)”, “Population Decrease Rate (PDR)”, “Birth Rate (BR)”, dan “Consumption Rate (CR)”. RGR merujuk pada tingkat kenaikan pertumbuhan dari sumber daya alam, PDR merujuk pada tingkat mortalitas populasi, BR merujuk pada tingkat kecepatan populasi berkembang biak, dan CR merujuk pada tingkat kecepatan sumber daya alam dikonsumsi.

Selain faktor-faktor diatas, ada juga beberapa asumsi yang perlu ditetapkan, seperti:

1. SDA akan selalu diproduksi.
2. Populasi akan selalu mengkonsumsi SDA.
3. Tidak ada migrasi penduduk.
4. Semua SDA berasal dari wilayah yang diobservasi.
5. Semua kematian disebabkan alasan alamiah.
6. Semua nilai parameter positif dan Real.

$$\frac{dP}{dt} = rP - aPC$$

$$\frac{dC}{dt} = bPC - sC$$

** Keterangan:*
P = jumlah SDA
C = jumlah populasi
r = resource growth rate (RGR)
a = birth rate (BR)
s = population decrease rate (PDR)
b = consumption rate (CR)

Ⅲ. Solusi Analitik

Karena sistem kedua persamaan tersebut tidak linear, maka kami tidak bisa menemukan solusi analitiknya. Namun, kami bisa menarik kesimpulan analitik, dengan melihat kestabilan dari .. di mana jika stabil, berarti populasi dan sumber daya akan bertemu di suatu titik secara terus menerus.

Untuk mencari kestabilan tersebut, kita bisa membuat agar masing-masing persamaan dibuat menjadi 0:

$$\frac{dP}{dt} = rP - aPC = f_1 = 0$$

$$\frac{dC}{dt} = bPC - sC = f_2 = 0$$

Di sini, perlu dicari terlebih dahulu *fixed points* di mana persamaan di atas memenuhi, yaitu pada saat:

$$P(r - aC) = 0$$
$$C(bP - s) = 0$$

Maka fixed point ada pada:

1) $P = 0$ dan $C = 0$ ya ng membuat fixed point berada pada $(0, 0)$

2) $P = 0$ dan $bP - s = 0$ yang membuat fixed point berada pada $(0, 0)$
3) $r - aC = 0$ dan $C = 0$ yang membuat fixed point berada pada $(0, 0)$
4) $r - aC = 0$ dan $C = 0$

Perlu juga dicari turunan parsial dari masing-masing persamaan.

$$f_1 = rP - aPC$$

$$\partial_P f_1 = r - aC \quad \partial_C f_1 = -aP$$

$$f_2 = bPC - sC$$

$$\partial_P f_1 = bC \quad \partial_C f_1 = bP - s$$

Maka:

$$A = \begin{bmatrix} r-aC & -aP \\ bC & bP-s \end{bmatrix}_{x \to a}$$

Coba masukkan fixed point (0,0) menjadi:

$$A = \begin{bmatrix} r & 0 \\ 0 & -s \end{bmatrix}$$

Mencari eigenvalues:

$$det(\begin{bmatrix} r & 0 \\ 0 & -s \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}) = 0$$

$$det(\begin{bmatrix} r-\lambda & 0 \\ 0 & -s-\lambda \end{bmatrix}) = 0$$

$$\lambda^2 + \lambda s - r\lambda - rs = 0$$

$$\lambda^2 + (s - r)\lambda - rs = 0$$

$$\lambda_1 = -s$$

$$\lambda_2 = r$$

ALTERNATIF

Dari (1):

$$P' = rP - bC$$

$$P' - rP = -bC$$

$$-\frac{1}{b}P' + \frac{r}{b}P = C \ldots (3)$$

Dari (2):

$$\frac{dC}{dt} = rP - sC$$

$$-\frac{1}{b}P'' + \frac{r}{b}P' = rP + \frac{s}{b}P' - \frac{sr}{b}P$$

$$-\frac{1}{b}P'' + (\frac{r-s}{b})P' + (\frac{sr}{b} - r)P = 0$$

$$-\frac{1}{b}z^2 + (\frac{r-s}{b})z + (\frac{sr}{b} - r) = 0$$

$$z_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{(\frac{r-s}{b}) \pm \sqrt{(\frac{r-s}{b})^2 - 4(-\frac{1}{b})(\frac{sr}{b} - r)}}{\frac{2}{b}}$$

Misalkan:
$r = 0.1$
$b = 0.01$
$s = 0.1$

Maka:

$$z_{1,2} = \frac{(\frac{0.1-0.1}{0.01}) \pm \sqrt{(\frac{0.1-0.1}{0.01})^2 - 4(-\frac{1}{0.01})(\frac{(0.1)(0.1)}{0.01} - 0.1)}}{\frac{2}{0.01}} = \frac{\pm\sqrt{360}}{200}$$

$$(z + \frac{\sqrt{360}}{200})(z - \frac{\sqrt{360}}{200}) = 0$$

$$P(t) = Ae^{\frac{\sqrt{360}}{200}t} + Be^{-\frac{\sqrt{360}}{200}t}$$

$$P'(t) = \frac{\sqrt{360}}{200}Ae^{\frac{\sqrt{360}}{200}t} - \frac{\sqrt{360}}{200}Be^{-\frac{\sqrt{360}}{200}t}$$

Subtitusikan ke (3):

$$C = -\frac{1}{0.01}(\frac{\sqrt{360}}{200}Ae^{\frac{\sqrt{360}}{200}t} - \frac{\sqrt{360}}{200}Be^{-\frac{\sqrt{360}}{200}t}) + \frac{0.1}{0.01}(Ae^{\frac{\sqrt{360}}{200}t} + Be^{-\frac{\sqrt{360}}{200}t})$$

Ⅳ. Solusi Numerik

Kami menggunakan bahasa pemrograman Python untuk menyelesaikan sistem ODE ini secara numerikal. Kami memakai metode Runge-Kutta orde ke-4 untuk menyelesaikan proses integrasinya.
Berikut adalah penjelasan fitur-fitur dari kode yang kami rancang:

```python
# Model
def system(t, y, r, a, s, b):
    P, C = y
    dPdt = r * P - a * P * C
    dCdt = b * P * C - s * C
    return np.array([dPdt, dCdt])
```

Fungsi ini untuk mendefinisikan sistem ODE yang telah kami rancang.
Variabel P, C, r, a, s, dan b memiliki definisi yang sama seperti yang sebelumnya pernah dibahas.
Variabel t menyimpan nilai waktu.
Variabel y menyimpan nilai total populasi dan sumber daya alam saat waktu t.
Variabel dPdt menyimpan nilai tingkat perubahan sumber daya alam per satuan waktu.
Variabel dCdt menyimpan nilai tingkat perubahan populasi per satuan waktu.

```python
# Fourth-order Runge-Kutta
def runge_kutta(h, t, y, r, a, s, b):
    k1 = h * system(t, y, r, a, s, b)
    k2 = h * system(t + h / 2, y + k1 / 2, r, a, s, b)
    k3 = h * system(t + h / 2, y + k2 / 2, r, a, s, b)
    k4 = h * system(t + h, y + k3, r, a, s, b)
    return y + (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

Fungsi ini untuk menyelesaikan ODE secara numerik dengan metode Runge-Kutta.
Variabel h menyimpan nilai step waktu.
Variabel k1, k2, k3, dan k4 menyimpan nilai estimasi turunan pada step tertentu.

```python
# Initial Populations
P0 = 40 # Initial Resource Available
C0 = 10 # Initial Total Population
initial_conditions = np.array([P0, C0])

# Parameters
r = 0.1  # Resource Growth Rate
a = 0.02  # Birth Rate
s = 0.1  # Population Decrease Rate
b = 0.01  # Consumption Rate
```

Bagian dari kode ini digunakan untuk mengatur nilai-nilai dari parameter yang ada.

```python
# Time Parameters
t_start = 0
t_end = 1000
num_points = 1000
h = (t_end - t_start) / num_points
```

Bagian dari kode ini digunakan untuk menentukan batas waktu awal dan akhir, berapa banyak poin di interval tersebut, dan berapa step waktu yang diperoleh.

```python
# Initialize arrays to store results
t_values = np.linspace(t_start, t_end, num_points)
solution_rk = np.zeros((num_points, 2))
```

Bagian dari kode ini digunakan untuk membuat variabel yang akan menyimpan nilai waktu dan perhitungan Runge-Kutta.

```python
# Runge-Kutta Calculations
solution_rk[0, :] = initial_conditions
for i in range(1, num_points):
    solution_rk[i, :] = runge_kutta(h, t_values[i-1], solution_rk[i-1, :], r, a, s, b)
```

Bagian dari kode ini digunakan untuk menggunakan fungsi runge_kutta() yang dibuat sebelumnya untuk menghitung nilai total populasi dan ketersediaan sumber daya alam dalam interval waktu yang ditetapkan sebelumnya.

```
# Plot
plt.figure(figsize=(16,9))
plt.plot(t_values, solution_rk[:, 0], label='Resource Available (P)')
plt.plot(t_values, solution_rk[:, 1], label='Total Population (C)')
plt.xlabel('Time')
plt.ylabel('Population Size')
plt.title('Model (Runge-Kutta)')
plt.legend(loc="upper left")
plt.show()
```

Bagian dari kode ini digunakan untuk memvisualisasikan hasil dari perhitungan-perhitungan sebelumnya.

V. Analisis Solusi Analitik dan Numerik

Dalam menganalisis solusi analitik, kami akan menjelaskan terlebih dahulu apa itu Stability Theory. Stability Theory membahas kestabilan solusi persamaan diferensial dan lintasan sistem dinamik di bawah gangguan kecil pada kondisi awal. Disini, Teori stabilitas berfokus pada dinamika populasi dari waktu ke waktu, terutama kecenderungan mereka untuk mempertahankan ukuran konstan atau berfluktuasi di sekitar keseimbangan tertentu. Dampak dari nilai eigen + adalah nilainya yang akan divergen, yang berarti jumlah populasi dan penduduk akan menurun/meningkat terus menerus. Sedangkan jika - maka nilainya akan konvergen, yang berarti jumlah populasi dan penduduk akan bertemu dalam 1 titik.
Jika sistem ini stabil, maka populasi dan sumber daya akan mencapai titik kesetimbangan, jika tidak stabil, maka populasi dan sumber daya akan terus meningkat/menurun tanpa batas waktu dengan eksponensial. Jika netral, maka penurunan/peningkatan tersebut akan terjadi lebih linear.

Untuk solusi numerik, kami akan melakukan simulasi perhitungan.
Misalkan:
Sumber daya awal (P0) adalah 40
Populasi awal (C0) adalah 10
Pertumbuhan sumber daya (r) adalah 0.1
Angka kelahiran/natalitas (a) adalah 0.02
Angka kematian/mortalitas (s) adalah 0.1
Konsumsi sumber daya (b) adalah 0.01

Disini, array inital_condition akan menjadi [40,10]
Untuk time parameter karena kami hanya akan simulasikan 1 titik, maka ini tidak akan kita simulasikan.

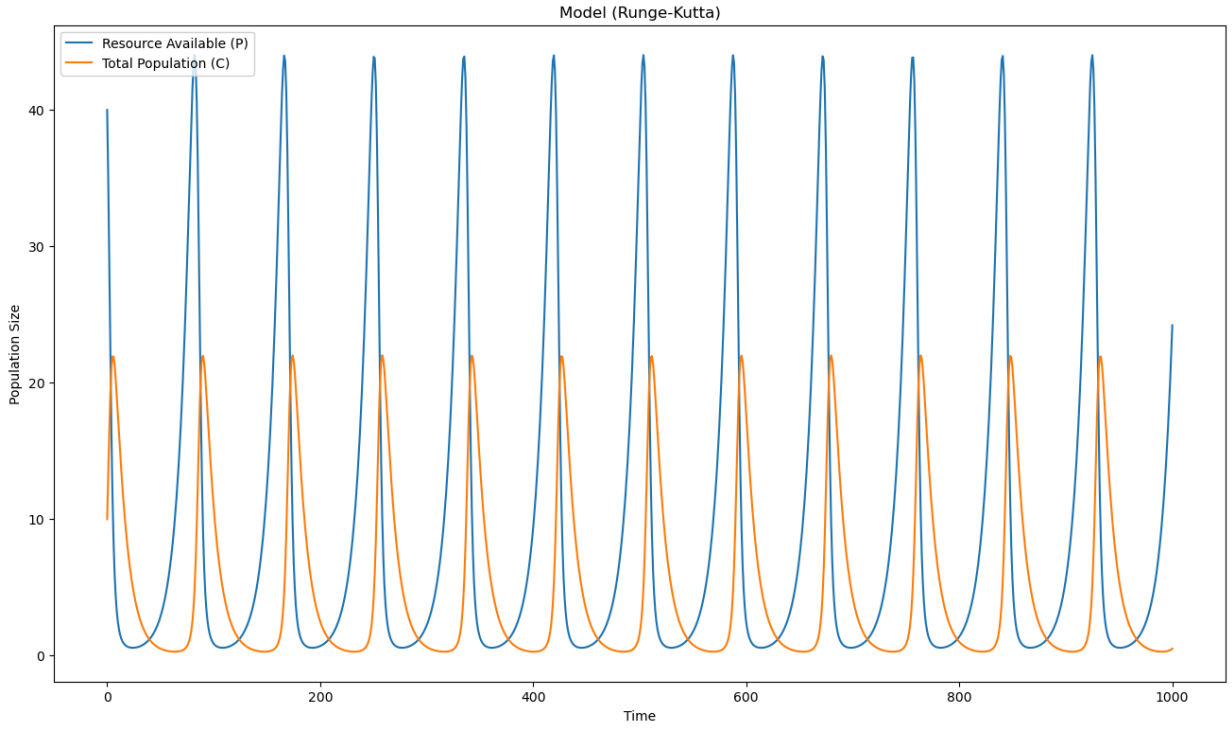
Sekarang kami akan melakukan simulasi runge_kutta dari (h, t_values[i-1], solution_rk[i-1,:], r,a,s,b)
h, yaitu perubahan waktu, yang akan dimisalkan 1
t_values adalah waktu, yang akan dimisalkan 10
Solution_rk merupakan hasil dari penghitungan, namun di argumen ini, solution_rk [i-1,:] menandakan bahwa itu hasil dari solusi sebelumnya, disini akan dimisalkan nilai ini [40,10] (sesuai dengan initial condition)
Sedangkan r,a,s,b sudah sebelumnya dimisalkan.

Lalu dengan kode yang telah diberikan sebelumnya, inilah hasil simulasi dari pengerjaannya:

Sekarang mengerjakan K1
P= 40

C= 10
dPdt= -4.0
dCdt= 3.0
k1= [-4. 3.]
Sekarang mengerjakan K2
P= 38.0
C= 11.5
dPdt= -4.9399999999999995
dCdt= 3.2199999999999998
k2= [-4.94 3.22]
Sekarang mengerjakan K3
P= 37.53
C= 11.61
dPdt= -4.961466
dCdt= 3.196233
k3= [-4.961466 3.196233]
Sekarang mengerjakan K4
P= 35.038534
C= 13.196233
dPdt= -5.743679772848438
dCdt= 3.3041432864242193
k4= [-5.74367977 3.30414329]
K1 hingga K4 telah ditemukan, sekarang menghitung hasil akhir
Hasil akhir adalah [35.0755647 13.18943488]
[35.0755647 13.18943488]

Hasil simulasi ini hanya akan menentukan 1 kondisi didalam 1 keadaan waktu itu saja, jika kita melakukannya selama 1000 satuan waktu, maka kami mendapatkan hasil sebagai berikut.



Disini, garis biru melambangkan sumber daya dan garis oranye melambangkan total populasi. Disini terdapat korelasi yaitu jika populasi meningkat, maka sumber daya akan menurun, dan jika populasi

menurun, maka sumber daya akan meningkat. Disini terlihat juga sebuah pola kejadian yang berulang dengan urutan sebagai berikut:
Populasi meningkat → Konsumsi meningkat → Sumber daya menurun → Sumber daya habis → Populasi menurun → Konsumsi menurun → Populasi habis → Sumber daya meningkat → Populasi meningkat, dan seterusnya.

Ⅵ. Kesimpulan

Solusi yang telah kami buat dapat melakukan simulasi terhadap bagaimana pertumbuhan penduduk dipengaruhi dengan berbagai faktor yaitu natalitas, mortalitas, dan bagaimana konsumsi sumber daya dan pertumbuhannya dengan baik. Baik dari solusi analitik maupun numerik. Kami juga dapat melakukan simulasi terhadap bagaimana jika salah satu dari 2 faktor penting yaitu penduduk dan sumber daya menjadi tidak seimbang, yaitu salah satu akan meningkat/menurun secara signifikan.

Ⅶ. Referensi

Caronongan, K. (2010) An Application of Differential Equations in the Study of Elastic Columns. Research Papers, Paper 5. http://opensiuc.lib.siu.edu/gs_rp/5

Ⅷ. Lampiran

Kode yang dipakai:

```
import numpy as np
import matplotlib.pyplot as plt

# Model
def system(t, y, r, a, s, b):
    P, C = y
    dPdt = r * P - a * P * C
    dCdt = b * P * C - s * C
    return np.array([dPdt, dCdt])

# Fourth-order Runge-Kutta
def runge_kutta(h, t, y, r, a, s, b):
    k1 = h * system(t, y, r, a, s, b)
    k2 = h * system(t + h / 2, y + k1 / 2, r, a, s, b)
    k3 = h * system(t + h / 2, y + k2 / 2, r, a, s, b)
    k4 = h * system(t + h, y + k3, r, a, s, b)
    return y + (k1 + 2 * k2 + 2 * k3 + k4) / 6

# Initial Populations
P0 = 40 # Initial Resource Available
C0 = 10 # Initial Total Population
initial_conditions = np.array([P0, C0])

# Parameters
```

```
r = 0.1  # Resource Growth Rate
a = 0.02  # Birth Rate
s = 0.1  # Population Decrease Rate
b = 0.01  # Consumption Rate

# Time Parameters
t_start = 0
t_end = 1000
num_points = 1000
h = (t_end - t_start) / num_points

# Initialize arrays to store results
t_values = np.linspace(t_start, t_end, num_points)
solution_rk = np.zeros((num_points, 2))

# Runge-Kutta Calculations
solution_rk[0, :] = initial_conditions
for i in range(1, num_points):
    solution_rk[i, :] = runge_kutta(h, t_values[i-1], solution_rk[i-1, :], r, a,
s, b)

# Plot
plt.figure(figsize=(16,9))
plt.plot(t_values, solution_rk[:, 0], label='Resource Available (P)')
plt.plot(t_values, solution_rk[:, 1], label='Total Population (C)')
plt.xlabel('Time')
plt.ylabel('Population Size')
plt.title('Model (Runge-Kutta)')
plt.legend(loc="upper left")
plt.show()
```