KONTROL *PREDATOR-PREY CHEMOSTAT*

(Tugas Akhir MK. Matematika Sistem)

Disusun Oleh:

Mohammad Ghani	**(1213201013)**
Wayan Rumite	**(1213201037)**
Nasruddin	**(1213201047)**

Dosen Pengampu:

Dr. Mardlijah, M.T.

PROGRAM PASCA MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKHNOLOGI SEPULUH NOPEMBER
SURABAYA
2014

KATA PENGANTAR

Segala puji bagi Allah SWT, Tuhan Yang Maha Esa. Yang telah senantiasa memberikan keberkahan, kesehatan dan rahmatnya kepada kita sekalian, sehinnga penulis dapat menyelesaikan laporan ini meskipun dalam bentuk yang sederhana. Shalawat serta salam semoga selalu tercurahkan kepada Nabi Muhammad SAW sebagai pelopor keberhasilan manusia dalam bertindak dan melaksanakan tugas.

Laporan ini sengaja disusun sebagai salah satu bentuk tugas wajib mata kuliah Matematika Sistem untuk dijadikan sebagai bahan referensi bagi penulis pada khususnya dan bagi pembaca pada umumnya sehingga diharapkan dapat menambah wawasan, keterampilan dan disiplin ilmu yang kita miliki agar dapat bermanfaat diberbagi aspek kepentingan umum terkhusus yang terkait dengan matakuliah tersebut.

Pada kesempatan ini, penulis menyampaikan banyak terimaksih kepada dosen pengampu mata kuliah Matematika Sistem oleh Dr. Mardlijah, MT yang telah banyak membantu penulis selama dalam proses perkulihan berlangsung. Seperti kata pepatah *"Tak ada gading yang tak retak"* begitupula dalam penulisan laporan ini, penulis menyadari bahwa masih terdapat kekurangan atau kekeliruan yang disebabkan oleh keterbatasan ilmu pengetahuan yang dimiliki. Oleh karena itu, penulis mengharapkan dukungan dan saran oleh pembaca yang sifatnya membangun demi penulisan berikutnya.

Surabaya,   Mei 2014

Penulis

DAFTAR ISI

KONTROL *PREDATOR-PREY CHEMOSTAT*

Abstrack

Untuk proyek ini, kita mendesain beberapa sistem kontrol untuk suatu predator-prey: suatu sistem chemostat yang diperoleh dengan menggunakan metode root locus dan memperhatikan tiga umpan balik, termasuk salah satunya menggunakan teknik kontrol optimal LQR. Kita juga mengukur kuantitas performance dari sistem tersebut dan ketahanan terhadap dan sensor suara. Kita menerapkan ketiga kontrol performance dan analisis ketahanan untuk kedua model linear dan nonlinear dari chemostat, dan menerapkan kontrol pengamatan untuk model nonlinear yang mencakup ketidak akuratan dalam parameter pengukuran, pengamatan tunda dan batasan laju dari penambahan nutrisi.

1. Sistem

Kita mendesai dan menganalisis berbagai sistem kontrol yang bervariasi untuk suatu sistem *predator-prey* pada lingkungan kimia yang statis, *chemostat* (pada Gambar 1). Nutrisi dialirkan kedalam *chemostat* dan dicampur dengan *soup organisme*: sampah akan dibersihkan untuk menjaga tingkat kecairan dalam wadah yang konstan.

Sistem ini terdiri atas *rotifera* (pemangsa) dengan konsentrasi $b(t)$ dan *Algae* (dimangsa) dengan konsentrasi $a\ (t)$, pada [1]. Kami membedakan konsentrasi nutrisi $n(t)$ dan mengontrol laju nutrisi yang ditambahkan ke dalam sistem $u(t)$.

Untuk model matematika diasumsikan sebagai berikut:

1) *Nutrisi* yang dicerna berbanding lurus dengan produk dari konsentrasi *nutrisi* dan *algae*.
2) Produksi ulang *algae* berbanding lurus dengan konsentrasi *nutrisi*, fungsi *algae* dimakan oleh *rotifera* berbanding lurus dengan konsentrasi yang dihasilkan oleh *algae*.
3) Produksi ulang dari *rotifera* berbanding lurus dengan laju *algae* yang di makan oleh *rotifera* dan *rotifera* mati ketika mengalami kerusakan *gen* kemudian menghasilkan nutrisi beracun.
4) Kita dapat mengukur konsentrasi dari *rotifera* dalam sistem dengan menggunakan alat ukur berupa *spectrofluorrimeter* [2].

Berdasarkan asumsi di atas dapat kitaiperoleh persamaan kendala dalam bentuk umum $\dot{x} = f(x,u)$

Dengan:

$$\dot{x} = \begin{pmatrix} \dot{n} \\ \dot{a} \\ \dot{b} \end{pmatrix} = \begin{pmatrix} u - k_1 na \\ k_1 \propto na - k_2 ab - k_3 a \\ k_2 \beta ab - k_4 nb \end{pmatrix} \tag{1}$$

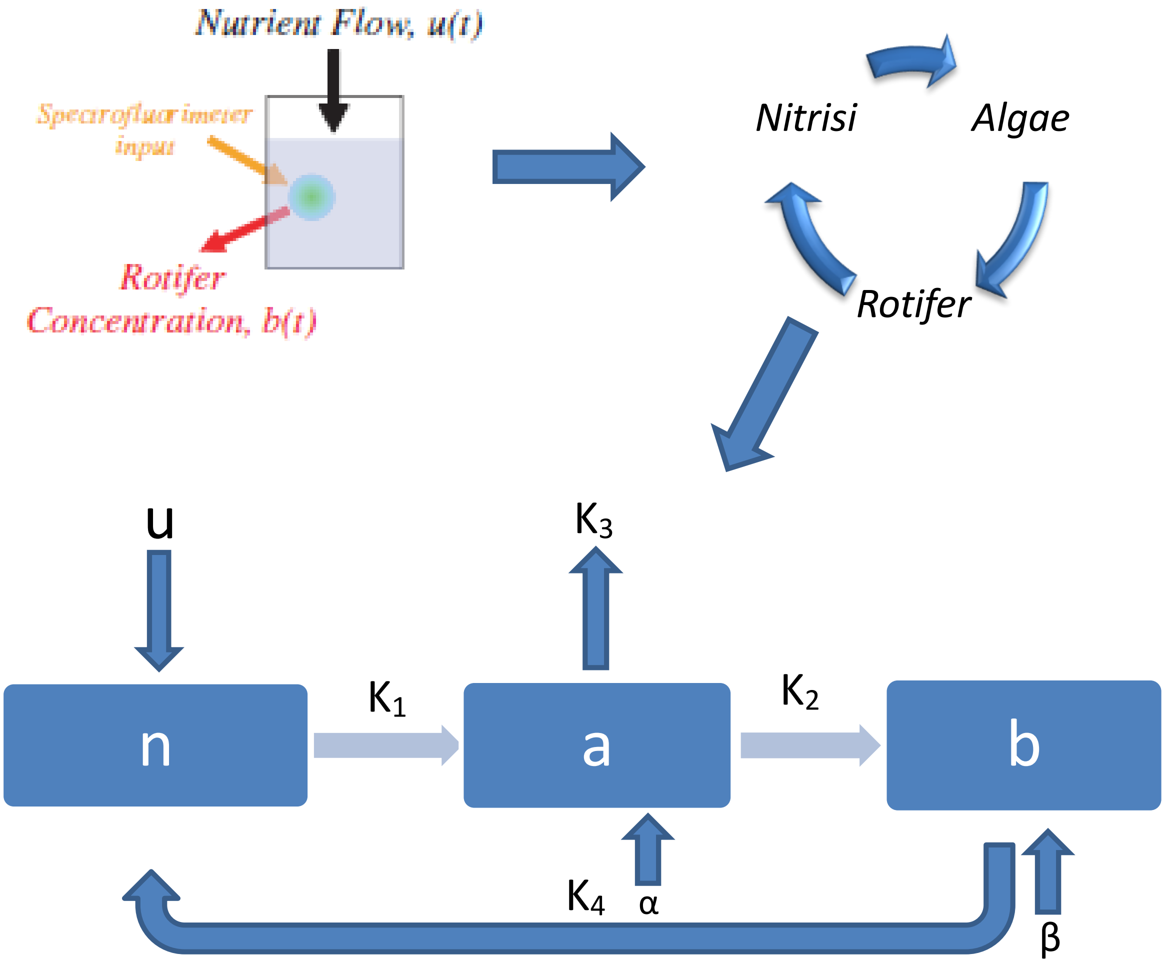
Misalkan:

$$f_1 = \dot{n} = u - k_1 na$$

$$f_2 = \dot{a} = k_1 \propto na - k_2 ab - k_3 a$$

$$f_3 = \dot{b} = k_2 \beta ab - k_4 nb$$

dan persamaan *output* $y = g_1 = b\,(t)$. (Diasumsikan konstan) parameter fisik yang diukur yaitu: $k_1 \approx 0{,}5$, $k_2 \approx 0.7$, $k_3 \approx 0{,}5$, $k_4 \approx 0.9$, $\alpha \approx 1.1$, $\beta \approx 2.0$. Kita dapat simulasikan sistem ini dengan menggunakan penambahan laju nutrisi *constant* $u(t) \equiv 1$.



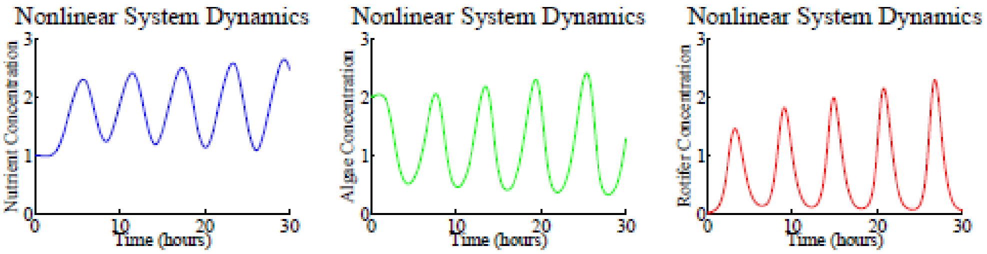


Fig. 2. Concentration of nutrients, algae, and rotifers versus time for the nonlinear system given by the equations in (1) with the initial condition n_0 = 2, a_0 = 1, b_0 = 0.01.

Gambar 2 menunjukkan konsentrasi *nutrisi*, *algae*, dan *rotifer* sebagai fungsi waktu.

Dinamika sistem yang mendasari menjadi jelas ketika kita memperhatikan bahwa konsentrasi *algae*, $a(t)$, kira-kira 180^0 keluar dari fase konsentrasi *rotifera*, seperti ilustrasi pada Gambar 3.

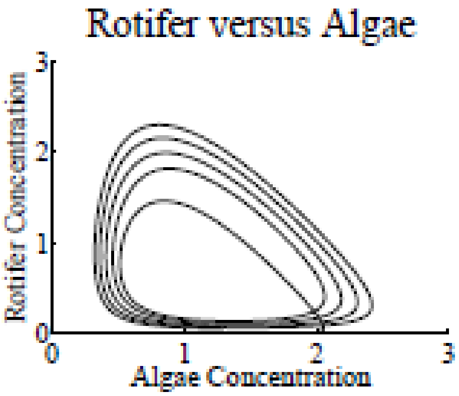


Fig. 3. Concentration of rotifer vs. concentration of algae over 30 hours. The two populations fluctuate out of phase, exhibiting one of the defining characteristics of a predator-prey system.

Ini berarti bahwa populasi maksimum *algae* kira-kira berbanding lurus dengan populasi minimum *rotifera*, dan begitupula sebaliknya. Hal inilah yang menggambarkan salah satu bentuk dari karakteristik sistem *prey-predator*.

Dalam *eksperiment* ini kita dapat melakukan kontrol terhadap *nutrisi*, *algae* dan *rotifera*. Percobaan ini menggunakan konsentrasi *konstant* $b \equiv 1$. Untuk mempertahankan kesetimbangan alam dalam hal ini dirancang *kompensator* untuk menghadapi gangguan "*impuls*", termasuk efek lingkungan seperti variasi suhu dan efek internal seperti evolusi dan kerusakan *nutrisi*.

Kita mulai dengan mendefinisikan satu set yang diinginkan kinerja spesifikasi-ifications, kemudian mengidentifikasi titik ekuilibrium sistem dan linearize sistem tentang hal itu. Lalu kita membuat baterai kompensator untuk sistem menggunakan root locus.

Perkembangan populasi *rotifera* merupakan ancaman bagi *algae* sehingga *algae* memberikan respon agar tidak disenangi olah *predator* [3]. Oleh karena itu, kestabilan dari konsentrasi *rotifera* pada 1,0 untuk mencegah perkembangan dari *algae*.

Untuk melakukan perkembangan kembali, *alga* memerlukan waktu rata-rata 5 jam. Hal ini dilakukan untuk menjaga perubahan dari model sistem yang ada dan untuk menjaga kestabilan populasi, *rotifera* membutuhkan waktu minimal 5 jam.

$$P_0 < 50\%, T_s < 5 \text{ jam} \tag{2}$$

Untuk lebih akurat, kita dapat juga menggunakan:

$$P_0 < \frac{i}{10}\%, T_s < 1 \text{ jam} \tag{3}$$

Jika kita mendekati sistem sebagai orde kedua, maka kita menempatkan dominan pada orde kedua pada $p \pm = -4 \pm 1{,}82j$.

1.1 Titik Kesetimbangan dari Sistem

Kita dapat mengontrol perilaku sistem ini dengan titik ekuilibrium yang ditetapkan sebagai $b \equiv 1$. Dari persamaan dalam (1) dengan $\dot{n} \equiv 0$, $\dot{a} \equiv 0$, dan $\dot{b} \equiv 0$ dengan pendekatan $b \equiv 1$.

$$x^* = \begin{pmatrix} n^* \\ a^* \\ b^* \\ u^* \end{pmatrix} = \begin{pmatrix} \frac{k_2+k_3}{\propto k_1} \\ \frac{(k_2+k_3)k_4}{\propto \beta k_1 k_2} \\ 1 \\ \frac{(k_2+k_3)^2 k_4}{\alpha^2 \beta k_2 k_2} \end{pmatrix} \tag{4}$$

2. Linearisasi Sistem

Dengan memperhatikan persamaan (1) yang memiliki bentuk $\dot{x} = f(x, u)$, kita mendapatkan sistem yang linear disekitar titik setimbang yang dapat dibentuk dalam matriks A, B dan C.

$$A = \frac{\partial f}{\partial x}\Big|_{x=x^*}^{u=u^*}, B = \frac{\partial f}{\partial u}\Big|_{x=x^*}^{u=^*}, C = \frac{\partial y}{\partial x}\Big|_{x=x^*}^{u=u^*} \tag{5}$$

dengan,

$$x_e = Ax_e + \dot{B}u_e, y_e = Cx_e \tag{6}$$

Dari bentuk umum pelinearan:

1) Untuk mencari matriks A.

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial n} & \frac{\partial f_1}{\partial a} & \frac{\partial f_1}{\partial b} \\ \frac{\partial f_2}{\partial n} & \frac{\partial f_2}{\partial a} & \frac{\partial f_2}{\partial b} \\ \frac{\partial f_3}{\partial n} & \frac{\partial f_3}{\partial a} & \frac{\partial f_3}{\partial b} \end{bmatrix}$$

$$A = \begin{bmatrix} -k_1 a & -k_1 n & 0 \\ k_1 \alpha a & k_1 n - k_2 b - k_3 & -k_2 a \\ -k_4 b & k_2 \beta b - k_4 n & k_2 \beta a - k_4 n \end{bmatrix}$$

$$A = \begin{bmatrix} -k_1\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1 k_2}\right) & -k_1\left(\frac{k_2+k_3}{\alpha k_1}\right) & 0 \\ k_1\alpha\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1 k_2}\right) & k_1\left(\frac{k_2+k_3}{\alpha k_1}\right) - k_2(1) - k_3 & -k_2\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1 k_2}\right) \\ -k_4(1) & k_2\beta(1) - k_4\left(\frac{k_2+k_3}{\alpha k_1}\right) & k_2\beta\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1 k_2}\right) - k_4\left(\frac{k_2+k_3}{\alpha k_1}\right) \end{bmatrix}$$

Sehinnga diperoleh matriks A sebagai berikut:

$$A = \begin{bmatrix} -\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_2}\right) & -\frac{k_2+k_3}{\alpha} & 0 \\ \frac{(k_2+k_3)k_4}{\alpha\beta k_2} & 0 & -\frac{(k_2+k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{bmatrix}$$

2) Untuk mencari Matriks B

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{\partial(u - k_1 na)}{\partial u} \\ \frac{\partial(k_1 \propto na - k_2 ab - k_3 a\,)}{\partial u} \\ \frac{\partial(k_2 \beta ab - k_4 nb\,)}{\partial u} \end{bmatrix}$$

Sehinnga diperoleh matriks B sebagai berikut:

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

3) Untuk mencari matriks C

$$C = \begin{bmatrix} \frac{\partial g_1}{\partial u} & \frac{\partial g_1}{\partial a} & \frac{\partial g_1}{\partial b} \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{\partial(b(t))}{\partial u} & \frac{\partial(b(t))}{\partial a} & \frac{\partial(b(t))}{\partial b} \end{bmatrix}$$

Sehinnga diperoleh matriks C sebagai berikut:

$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

3. Nilai Eigen

Berikut adalah matriks A:

$$A = \begin{bmatrix} -\left(\frac{(k_2 + k_3)k_4}{\alpha\beta k_2}\right) & -\frac{k_2 + k_3}{\alpha} & 0 \\ \frac{(k_2 + k_3)k_4}{\alpha\beta k_2} & 0 & -\frac{(k_2 + k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{bmatrix}$$

Dengan mensubstitusikan nilai dari parameter-parameter yang telah diberikan sebelumnya yaitu: $k_1 \approx 0{,}5$, $k_2 \approx 0.7$, $k_3 \approx 0{,}5$, $k_4 \approx 0.9$, $\alpha \approx 1.1$, $\beta \approx 2.0$.

Didapat nilai matriks A sebagai berikut:

$$A = \begin{bmatrix} -1.3745 & -1.0909 & 0 \\ 0.3780 & 0 & -0.9818 \\ -0.9000 & 1.4000 & 0 \end{bmatrix}$$

Dengan menggunakan bantuan program MATLAB diperoleh nilai aigen dari matriks A adalah sebagai berikut:

$$e = \begin{bmatrix} -1.4748 & 0 & 0 \\ 0 & 0.0501 + 1.3900i & 0 \\ 0 & 0 & 0.0501 - 1.3900i \end{bmatrix}$$

Dari nilai aigen yang diperoleh di atas tampak bahwa sistem tidak stabil, karena semua entri pada nilai aigen tidak imajiner, sehingga akan diberikan *feed-back* agar sistem terstabilkan.

4. Keteramatan Sistem

Suatu sistem dapat dikatakan teramati jika memenuhi:

$$rank\ (M_o) = rank\ (A)$$

$$A = \begin{bmatrix} -\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_2}\right) & -\frac{k_2+k_3}{\alpha} & 0 \\ \frac{(k_2+k_3)k_4}{\alpha\beta k_2} & 0 & -\frac{(k_2+k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{bmatrix}$$

Matriks A:

$$A = \begin{bmatrix} -1.3745 & -1.0909 & 0 \\ 0.3780 & 0 & -0.9818 \\ -0.9000 & 1.4000 & 0 \end{bmatrix}$$, memiliki rank = 3

Matrik C:

$C = [0 \quad 0 \quad 1]$

Matrik M_o secara umum sebagai berikut:

$$M_o = \begin{pmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

Karena A adalah matriks yang berordo 3 maka diperoleh matriks M_o sebagai berikut:

$M_o = \begin{bmatrix} 0 & 0 & 1 \\ -0.9000 & 1.4000 & 0 \\ 1.7663 & 0.9818 & -1.3745 \end{bmatrix}$, memiliki *rank* = 3

karena $rank\ (M_o) = rank\ (A)$, maka dapat disimpulkan bahwa sistem teramati.

5. Keterkontrolan Sistem

Suatu sistem dapat dikatakan Terkontrol jika memenuhi:

$$rank\ (M_c) = rank\ (A)$$

$$A = \begin{bmatrix} -\left(\frac{(k_2+k_3)k_4}{\alpha\beta k_2}\right) & -\frac{k_2+k_3}{\alpha} & 0 \\ \frac{(k_2+k_3)k_4}{\alpha\beta k_2} & 0 & -\frac{(k_2+k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{bmatrix}$$

Matriks A:

$A = \begin{bmatrix} -1.3745 & -1.0909 & 0 \\ 0.3780 & 0 & -0.9818 \\ -0.9000 & 1.4000 & 0 \end{bmatrix}$, memiliki *rank* = 3

Matrik B:

$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Matrik M_c secara umum sebagai berikut:

$$M_c = [B \quad AB \quad A^2B \quad A^3B \quad \dots \quad A^{n-1}B]$$

Karena A adalah matriks yang berordo 3 maka diperoleh matriks M_c sebagai berikut:

$M_c = \begin{bmatrix} 1 & -1.3745 & 1.4770 \\ 0 & -1.3745 & 0.3641 \\ 0 & -0.9000 & 1.7663 \end{bmatrix}$, memiliki *rank* = 3

karena $rank\ (M_c) = rank\ (A)$, maka dapat disimpulkan bahwa sistem terkontrol.

6. Kestabilan Sistem

- **Persamaan sistem di titik setimbang:**

$$x^* = \begin{bmatrix} n^* \\ a^* \\ b^* \end{bmatrix} = \begin{bmatrix} \frac{k_2 + k_3}{\alpha k_1} \\ \frac{(k_2 + k_3)k_4}{\alpha \beta k_1 k_2} \\ 1 \end{bmatrix}$$

- **Hasil pelinieran:**
Matriks Jacobian:

$$J(x^*) = \begin{bmatrix} -\frac{(k_2 + k_3)k_4}{\alpha\beta k_2} & -\frac{k_2 + k_3}{\alpha} & 0 \\ \frac{(k_2 + k_3)k_4}{\beta k_2} & 0 & -\frac{(k_2 + k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{bmatrix}$$

$$J(x^*) = \begin{bmatrix} a_1 & a_2 & 0 \\ a_3 & 0 & a_4 \\ a_5 & a_6 & 0 \end{bmatrix}$$

Dengan:

$$a_1 = -\frac{(k_2 + k_3)k_4}{\alpha\beta k_2}$$

$$a_2 = -\frac{k_2 + k_3}{\alpha}$$

$$a_3 = \frac{(k_2 + k_3)k_4}{\beta k_2}$$

$$a_4 = -\frac{(k_2 + k_3)k_4}{\alpha\beta k_1}$$

$$a_5 = -k_4$$

$$a_6 = \beta k_2$$

- **Kriteria Karakteristik:** $(\lambda I - A)x = 0$

$$\det\left(\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \begin{bmatrix} a_1 & a_2 & 0 \\ a_3 & 0 & a_4 \\ a_5 & a_6 & 0 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} \lambda - a_1 & -a_2 & 0 \\ -a_3 & \lambda & -a_4 \\ -a_5 & -a_6 & \lambda \end{bmatrix}\right) = 0$$

Didapat persamaan karakteristik sebagai berikut:

$$(\lambda - a_1)(\lambda)(\lambda) + (-a_2)(-a_4)(-a_5) - ((-a_4)(-a_6)(\lambda - a_1) + (-a_2)(-a_3)\lambda) = 0$$

$$\lambda^3 - a_1\lambda - a_2a_4a_5 - (a_4a_6\lambda - a_1a_4a_6 + a_2a_3\lambda) = 0$$

$$\lambda^3 - (a_1 + a_4a_6 + a_2a_3)\lambda - a_2a_4a_5 + a_1a_4a_6 = 0$$

$$\lambda^3 + B\lambda + C = 0$$

Dengan:

$$B = -(a_1 + a_4a_6 + a_2a_3)$$

$$C = -a_2a_4a_5 + a_1a_4a_6$$

❖ $B = -(a_1 + a_4a_6 + a_2a_3)$

$$= -\left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_2} + \left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)(\beta k_2) + \left(-\frac{k_2+k_3}{\alpha}\right)\left(\frac{(k_2+k_3)k_4}{\beta k_2}\right)\right)$$

$$= \frac{(k_2+k_3)k_4}{\alpha\beta k_2} + \left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)(\beta k_2) + \left(\frac{k_2+k_3}{\alpha}\right)\left(\frac{(k_2+k_3)k_4}{\beta k_2}\right)$$

karena $k_1, k_2, k_3, k_4, \alpha, \beta > 0$, maka $B > 0$

❖ $C = -a_2a_4a_5 + a_1a_4a_6$

$$= a_4(a_1a_6 - a_2a_5)$$

$$= \left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)\left[\left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_2}\right)(\beta k_2) - \left(\left(-\frac{k_2+k_3}{\alpha}\right)(-k_4)\right)\right]$$

$$= \left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)\left[\left(-\frac{(k_2+k_3)}{\alpha}\right)(k_4) - \left(\left(-\frac{k_2+k_3}{\alpha}\right)(-k_4)\right)\right]$$

$$= \left(-\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)\left[2\left(-\frac{(k_2+k_3)}{\alpha}\right)(k_4)\right]$$

$$= \left(\frac{(k_2+k_3)k_4}{\alpha\beta k_1}\right)\left[2\left(\frac{(k_2+k_3)}{\alpha}\right)(k_4)\right]$$

karena $k_1, k_2, k_3, k_4, \alpha, \beta > 0$, maka $C > 0$

❖ Syarat perlu dan syarat cukup kondisi untuk kesetabilan *asimtotic* dari kriteria Routh-Hurwith $AB - C > 0$. Karena $A = 0$ maka $AB - C = 0 - C > 0$ jika dan hanya jika $C < 0$. Sehengga, kriteria Routh-Hurwith tidak. Hal ini dapat menjadi masukkan untuk sistem dinamik *Predator-Prey* yang tidak stabil.

Program Simulasi

Source Code (GUI)

```
function varargout = UAS_MATSIS(varargin)
% UAS_MATSIS M-file for UAS_MATSIS.fig
%      UAS_MATSIS, by itself, creates a new UAS_MATSIS or
raises the existing
%      singleton*.
%
%      H = UAS_MATSIS returns the handle to a new UAS_MATSIS
or the handle to
%      the existing singleton*.
%
%      UAS_MATSIS('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in UAS_MATSIS.M with the given
input arguments.
%
%      UAS_MATSIS('Property','Value',...) creates a new
UAS_MATSIS or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before UAS_MATSIS_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to UAS_MATSIS_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
UAS_MATSIS

% Last Modified by GUIDE v2.5 26-May-2014 14:32:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @UAS_MATSIS_OpeningFcn,
...
                   'gui_OutputFcn',  @UAS_MATSIS_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before UAS_MATSIS is made visible.
function UAS_MATSIS_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to UAS_MATSIS (see
VARARGIN)

% Choose default command line output for UAS_MATSIS
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes UAS_MATSIS wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = UAS_MATSIS_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function n0_Callback(hObject, eventdata, handles)
% hObject    handle to n0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of n0 as text
%        str2double(get(hObject,'String')) returns contents of
n0 as a double


% --- Executes during object creation, after setting all
properties.
function n0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function a0_Callback(hObject, eventdata, handles)
% hObject    handle to a0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of a0 as text
%        str2double(get(hObject,'String')) returns contents of
a0 as a double
```

```
% --- Executes during object creation, after setting all
properties.
function a0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function b0_Callback(hObject, eventdata, handles)
% hObject    handle to b0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of b0 as text
%        str2double(get(hObject,'String')) returns contents of
b0 as a double


% --- Executes during object creation, after setting all
properties.
function b0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b0 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

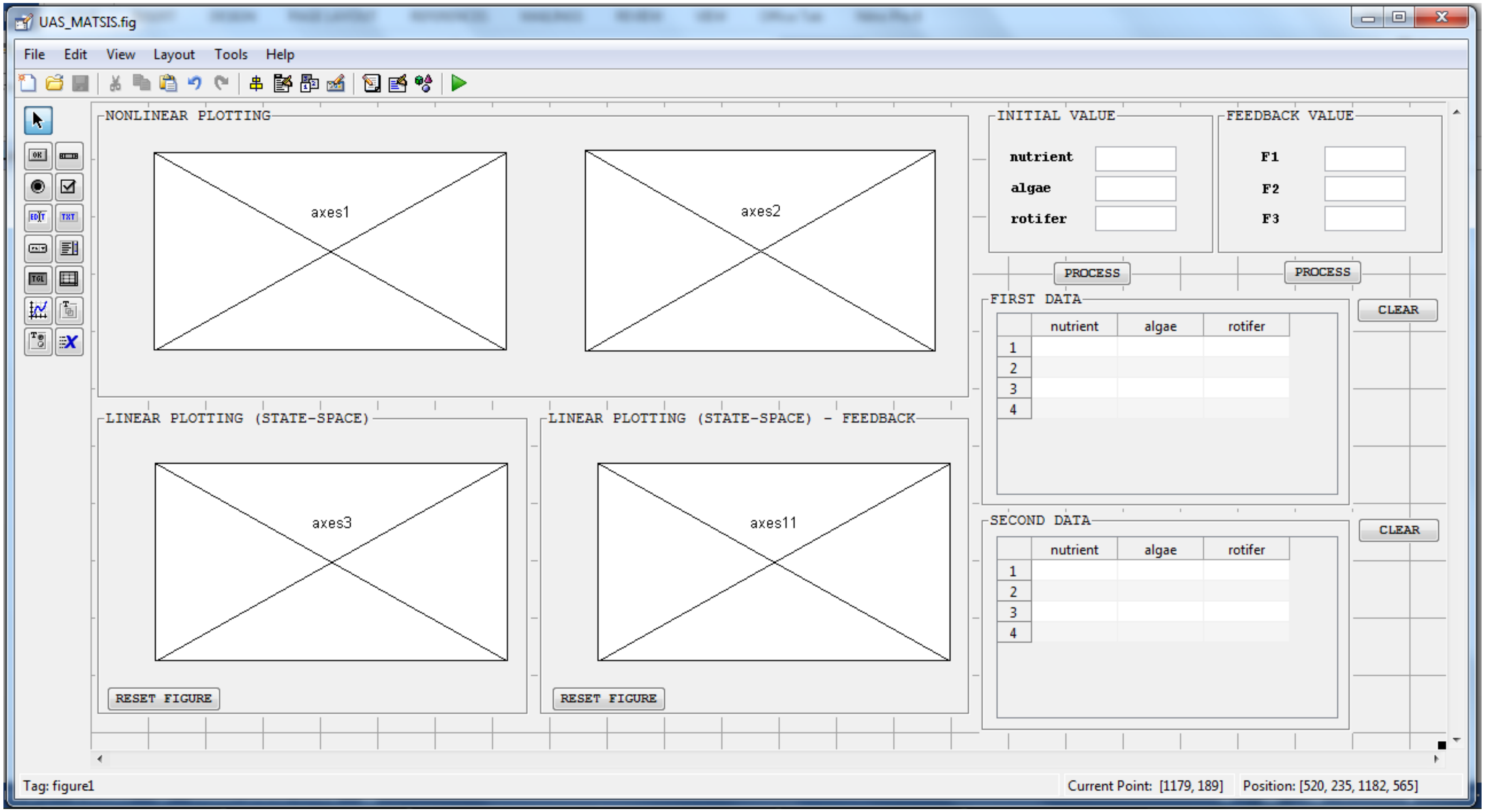
```
% --- Executes on button press in process1.
function process1_Callback(hObject, eventdata, handles)
% hObject    handle to process1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;
k1=0.5;k2=0.7;k3=0.5;k4=0.9;alpha=1.1;beta=2;
A=[(-1)*(k2+k3)*k4/alpha*beta*k2 (-1)*(k2+k3)/alpha 0;...
     (k2+k3)*k4/beta*k2 0 (-1)*(k2+k3)*k4/alpha*beta*k1;...
    -k4 beta*k2 0
    ];
B=[1;0;0];
C=[0 0 1];
%% NONLINEAR STEP
h=0.2;
t=0:h:30;
% n(1)=2;a(1)=1;b(1)=0.01;
handles.x=str2num(get(handles.n0,'String'));
handles.y=str2num(get(handles.a0,'String'));
handles.z=str2num(get(handles.b0,'String'));
n(1)=handles.x;a(1)=handles.y;b(1)=handles.z;
for i=1:length(t)-1
    u(i)=1;
    n1=h*(u(i)-k1*n(i)*a(i));
    a1=h*(k1*alpha*n(i)*a(i)-k2*a(i)*b(i)-k3*a(i));
    b1=h*(k2*beta*a(i)*b(i)-k4*n(i)*b(i));

    n2=h*(u(i)-k1*(n(i)+0.5*n1)*(a(i)+0.5*a1));
    a2=h*(k1*alpha*(n(i)+0.5*n1)*(a(i)+0.5*a1)-
k2*(a(i)+0.5*a1)*(b(i)+0.5*b1)-k3*(a(i)+0.5*a1));
    b2=h*(k2*beta*(a(i)+0.5*a1)*(b(i)+0.5*b1)-
k4*(n(i)+0.5*n1)*(b(i)+0.5*b1));

    n3=h*(u(i)-k1*(n(i)+0.5*n2)*(a(i)+0.5*a2));
    a3=h*(k1*alpha*(n(i)+0.5*n2)*(a(i)+0.5*a2)-
k2*(a(i)+0.5*a2)*(b(i)+0.5*b2)-k3*(a(i)+0.5*a2));
    b3=h*(k2*beta*(a(i)+0.5*a2)*(b(i)+0.5*b2)-
k4*(n(i)+0.5*n2)*(b(i)+0.5*b2));

    n4=h*(u(i)-k1*(n(i)+n3)*(a(i)+a3));
    a4=h*(k1*alpha*(n(i)+n3)*(a(i)+a3)-k2*(a(i)+a3)*(b(i)+b3)-
k3*(a(i)+a3));
    b4=h*(k2*beta*(a(i)+a3)*(b(i)+b3)-k4*(n(i)+n3)*(b(i)+b3));

    n(i+1)=n(i)+(1/6)*(n1+2*n2+2*n3+n4);
    a(i+1)=a(i)+(1/6)*(a1+2*a2+2*a3+a4);
    b(i+1)=b(i)+(1/6)*(b1+2*b2+2*b3+b4);
end;
axes(handles.axes1);
```

```
plot(t,n,'b');
hold on;
plot(t,a,'g');
hold on;
plot(t,b,'r');
xlabel('Time (hours)');
ylabel('Concentration');
legend('Nutrient','Algae','Rotifer');
axes(handles.axes2);
plot(a,b,'m');
title('Rotifer Versus Algae');
xlabel('Algae Concentration');
ylabel('Rotifer Concentration');
%% LINEAR STEP
h_linear=0.2;
t_linear=0:h_linear:50;
A1=-(k2+k3)*k4/(alpha*beta*k2);
A2=-(k2+k3)/alpha;
A3=(k2+k3)*k4/(beta*k2);
A4=-(k2+k3)*k4/(alpha*beta*k1);
A5=-k4;
A6=beta*k2;
n_star=(k2+k3)/alpha*k1;
a_star=(k2+k3)*k4/alpha*beta*k1*k2;
b_star=1;
u_star=((k2+k3)^2)*k4/(alpha^2)*beta*(k2^2);
n_linear(1)=handles.x;a_linear(1)=handles.y;b_linear(1)=handle
s.z;
xe1(1)=n_linear(1)-n_star;
xe2(1)=a_linear(1)-a_star;
xe3(1)=b_linear(1)-b_star;
for i=1:length(t_linear)-1
    u(i)=1;
    ue(i)=u(i)-u_star;
    xe1_value=h_linear*(A1*xe1(i)+A2*xe2(i)+ue(i));
    xe2_value=h_linear*(A3*xe1(i)+A4*xe3(i));
    xe3_value=h_linear*(A5*xe1(i)+A6*xe2(i));


xe1_value2=h_linear*(A1*(xe1(i)+0.5*xe1_value)+A2*(xe2(i)+0.5*
xe2_value)+ue(i));

xe2_value2=h_linear*(A3*(xe1(i)+0.5*xe1_value)+A4*(xe3(i)+0.5*
xe3_value));

xe3_value2=h_linear*(A5*(xe1(i)+0.5*xe1_value)+A6*(xe2(i)+0.5*
xe2_value));


xe1_value3=h_linear*(A1*(xe1(i)+0.5*xe1_value2)+A2*(xe2(i)+0.5
*xe2_value2)+ue(i));
```

```
xe2_value3=h_linear*(A3*(xe1(i)+0.5*xe1_value2)+A4*(xe3(i)+0.5
*xe3_value2));

xe3_value3=h_linear*(A5*(xe1(i)+0.5*xe1_value2)+A6*(xe2(i)+0.5
*xe2_value2));


xe1_value4=h_linear*(A1*(xe1(i)+xe1_value3)+A2*(xe2(i)+xe2_val
ue3)+ue(i));

xe2_value4=h_linear*(A3*(xe1(i)+xe1_value3)+A4*(xe3(i)+xe3_val
ue3));

xe3_value4=h_linear*(A5*(xe1(i)+xe1_value3)+A6*(xe2(i)+xe2_val
ue3));


xe1(i+1)=xe1(i)+(1/6)*(xe1_value+2*xe1_value2+2*xe1_value3+xe1
_value4);

xe2(i+1)=xe2(i)+(1/6)*(xe2_value+2*xe2_value2+2*xe2_value3+xe2
_value4);

xe3(i+1)=xe3(i)+(1/6)*(xe3_value+2*xe3_value2+2*xe3_value3+xe3
_value4);

    n_linear(i+1)=xe1(i)+n_star;
    a_linear(i+1)=xe2(i)+a_star;
    b_linear(i+1)=xe3(i)+b_star;
end;
axes(handles.axes3);
plot(t_linear,n_linear,'b');
hold on;
plot(t_linear,a_linear,'g');
hold on;
plot(t_linear,b_linear,'r');
xlabel('Time (hours)');
ylabel('Concentration');
legend('Nutrient','Algae','Rotifer');
DATA1=[n_linear' a_linear' b_linear'];
set(handles.data1,'data',DATA1);
guidata(hObject, handles);



function f1_Callback(hObject, eventdata, handles)
% hObject    handle to f1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of f1 as text
%        str2double(get(hObject,'String')) returns contents of
f1 as a double


% --- Executes during object creation, after setting all
properties.
function f1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function f2_Callback(hObject, eventdata, handles)
% hObject    handle to f2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of f2 as text
%        str2double(get(hObject,'String')) returns contents of
f2 as a double


% --- Executes during object creation, after setting all
properties.
function f2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function f3_Callback(hObject, eventdata, handles)
% hObject    handle to f3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of f3 as text
%        str2double(get(hObject,'String')) returns contents of
f3 as a double


% --- Executes during object creation, after setting all
properties.
function f3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in feedback.
function feedback_Callback(hObject, eventdata, handles)
% hObject    handle to feedback (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;
k1=0.5;k2=0.7;k3=0.5;k4=0.9;alpha=1.1;beta=2;
n_star=(k2+k3)/alpha*k1;
a_star=(k2+k3)*k4/alpha*beta*k1*k2;
b_star=1;
u_star=((k2+k3)^2)*k4/(alpha^2)*beta*(k2^2);
h_linear=0.2;
t_linear=0:h_linear:50;
```

```
A1=-(k2+k3)*k4/(alpha*beta*k2);
A2=-(k2+k3)/alpha;
A3=(k2+k3)*k4/(beta*k2);
A4=-(k2+k3)*k4/(alpha*beta*k1);
A5=-k4;
A6=beta*k2;
n_fb(1)=handles.x;a_fb(1)=handles.y;b_fb(1)=handles.z;
xe1_fb(1)=n_fb(1)-n_star;
xe2_fb(1)=a_fb(1)-a_star;
xe3_fb(1)=b_fb(1)-b_star;
% a_feedback=-5;
% b_feedback=-1;
% c_feedback=1;
a_feedback=str2num(get(handles.f1,'String'));
b_feedback=str2num(get(handles.f2,'String'));
c_feedback=str2num(get(handles.f3,'String'));
for i=1:length(t_linear)-1

xe1_aksen=h_linear*((A1+a_feedback)*xe1_fb(i)+(A2+b_feedback)*
xe2_fb(i)+c_feedback*xe3_fb(i));
   xe2_aksen=h_linear*(A3*xe1_fb(i)+A4*xe3_fb(i));
   xe3_aksen=h_linear*(A5*xe1_fb(i)+A6*xe2_fb(i));


xe1_aksen2=h_linear*((A1+a_feedback)*(xe1_fb(i)+0.5*xe1_aksen)
+(A2+b_feedback)*(xe2_fb(i)+0.5*xe2_aksen)+c_feedback*(xe3_fb(
i)+0.5*xe3_aksen));

xe2_aksen2=h_linear*(A3*(xe1_fb(i)+0.5*xe1_aksen)+A4*(xe3_fb(i
)+0.5*xe3_aksen));

xe3_aksen2=h_linear*(A5*(xe1_fb(i)+0.5*xe1_aksen)+A6*(xe2_fb(i
)+0.5*xe2_aksen));


xe1_aksen3=h_linear*((A1+a_feedback)*(xe1_fb(i)+0.5*xe1_aksen2
)+(A2+b_feedback)*(xe2_fb(i)+0.5*xe2_aksen2)+c_feedback*(xe3_f
b(i)+0.5*xe3_aksen2));

xe2_aksen3=h_linear*(A3*(xe1_fb(i)+0.5*xe1_aksen2)+A4*(xe3_fb(
i)+0.5*xe3_aksen2));

xe3_aksen3=h_linear*(A5*(xe1_fb(i)+0.5*xe1_aksen2)+A6*(xe2_fb(
i)+0.5*xe2_aksen2));


xe1_aksen4=h_linear*((A1+a_feedback)*(xe1_fb(i)+xe1_aksen3)+(A
2+b_feedback)*(xe2_fb(i)+xe2_aksen3)+c_feedback*(xe3_fb(i)+xe3
_aksen3));

xe2_aksen4=h_linear*(A3*(xe1_fb(i)+xe1_aksen3)+A4*(xe3_fb(i)+x
e3_aksen3));

xe3_aksen4=h_linear*(A5*(xe1_fb(i)+xe1_aksen3)+A6*(xe2_fb(i)+x
e2_aksen3));


xe1_fb(i+1)=xe1_fb(i)+(1/6)*(xe1_aksen+2*xe1_aksen2+2*xe1_akse
n3+xe1_aksen4);

xe2_fb(i+1)=xe2_fb(i)+(1/6)*(xe2_aksen+2*xe2_aksen2+2*xe2_akse
n3+xe2_aksen4);

xe3_fb(i+1)=xe3_fb(i)+(1/6)*(xe3_aksen+2*xe3_aksen2+2*xe3_akse
n3+xe3_aksen4);

   n_fb(i+1)=xe1_fb(i+1)+n_star;
   a_fb(i+1)=xe2_fb(i+1)+a_star;
   b_fb(i+1)=xe3_fb(i+1)+b_star;
end;
axes(handles.axes11);
plot(t_linear,n_fb,'b');
hold on;
plot(t_linear,a_fb,'g');
hold on;
plot(t_linear,b_fb,'r');
xlabel('Time (hours)');
ylabel('Concentration');
DATA2=[n_fb' a_fb' b_fb'];
set(handles.data2,'data',DATA2);
legend('Nutrient','Algae','Rotifer');


% --- Executes on button press in reset1.
function reset1_Callback(hObject, eventdata, handles)
% hObject    handle to reset1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;
axes(handles.axes1);
cla reset;
axes(handles.axes2);
cla reset;
axes(handles.axes3);
cla reset;


% --- Executes on button press in reset2.
function reset2_Callback(hObject, eventdata, handles)
% hObject    handle to reset2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;
axes(handles.axes11);
cla reset;

% --- Executes on button press in clear1.
function clear1_Callback(hObject, eventdata, handles)
% hObject    handle to clear1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.data1,'data',[]);

% --- Executes on button press in clear2.
function clear2_Callback(hObject, eventdata, handles)
% hObject    handle to clear2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.data2,'data',[]);
```

Hasil Program Simulasi



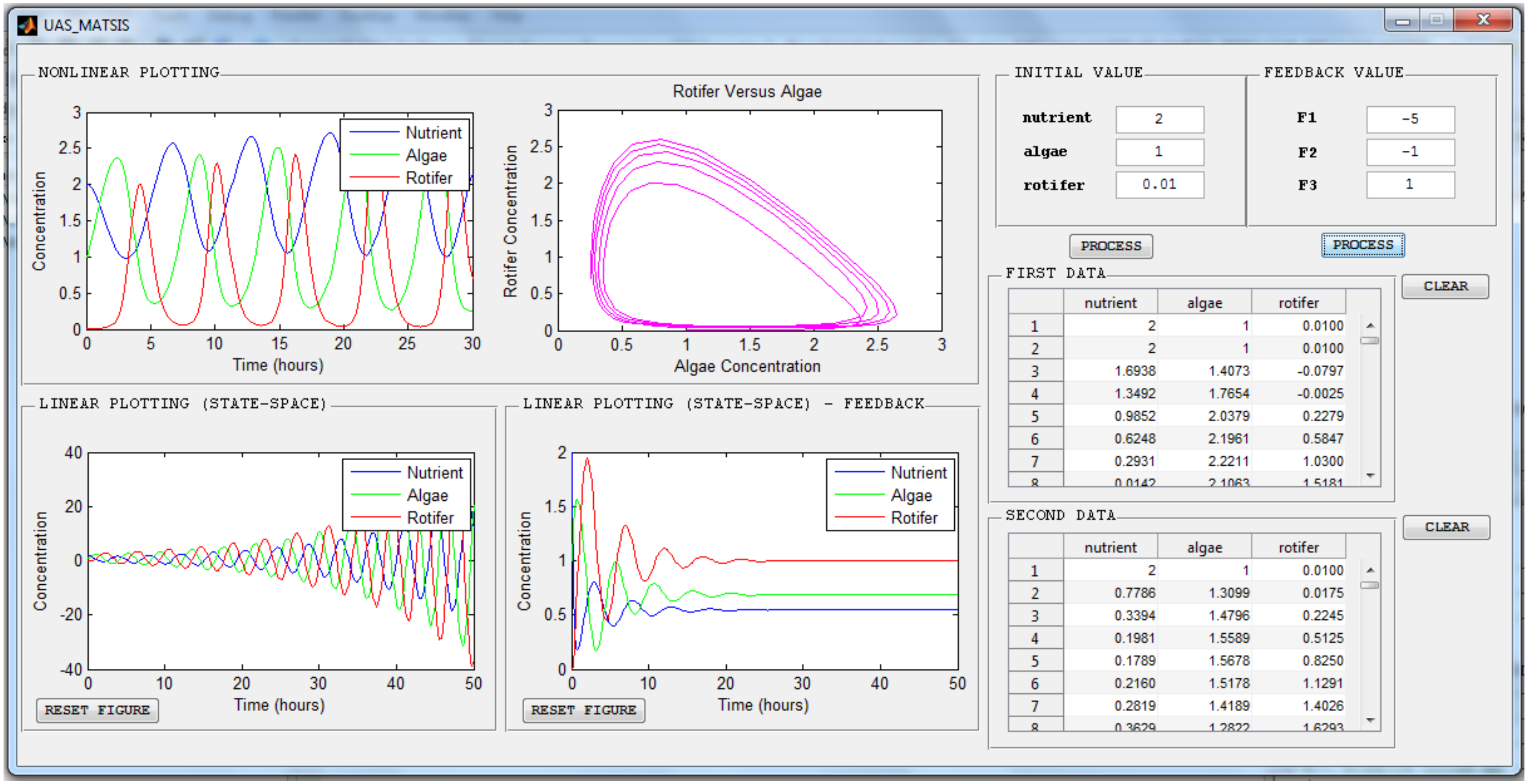
FIRST DATA

	nutrient	algae	rotifer
1			
2			
3			
4			

SECOND DATA

	nutrient	algae	rotifer
1			
2			
3			
4			

Tag: figure1 Current Point: [1179, 189] Position: [520, 235, 1182, 565]



INITIAL VALUE

nutrient	2
algae	1
rotifer	0.01

FEEDBACK VALUE

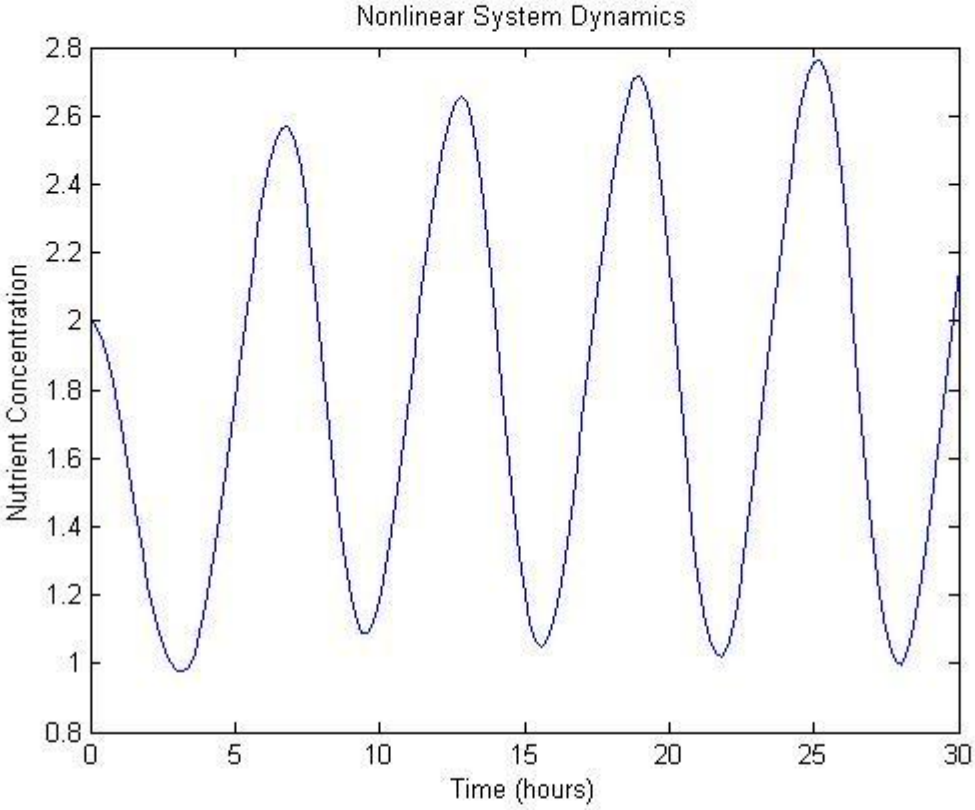
F1	-5
F2	-1
F3	1

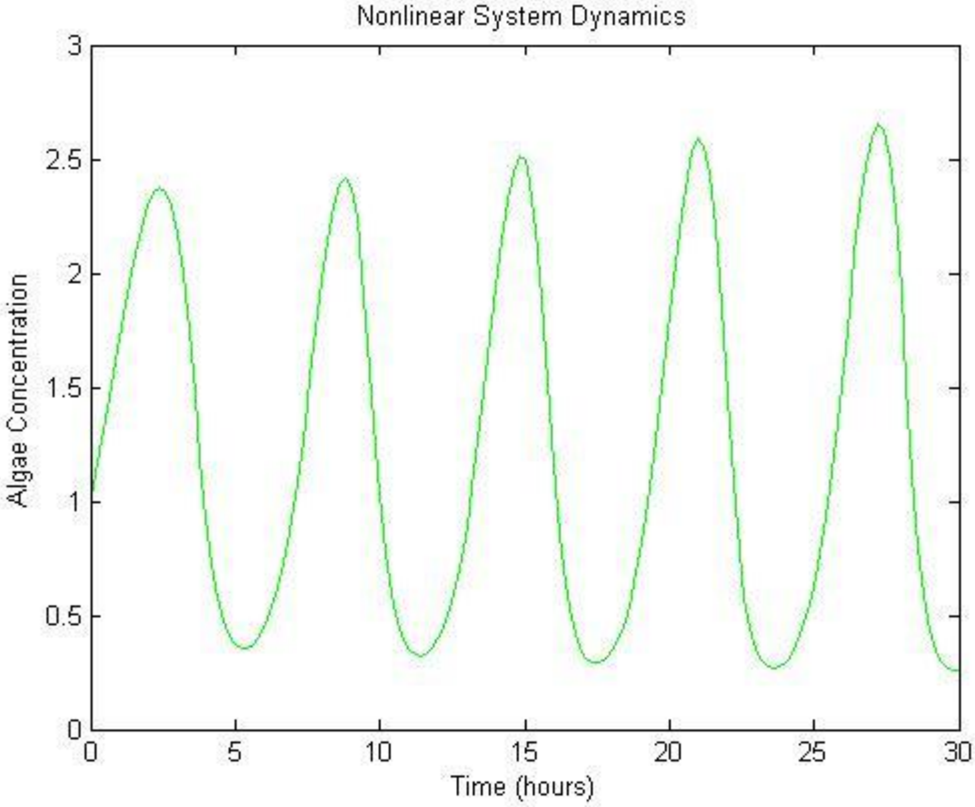
FIRST DATA

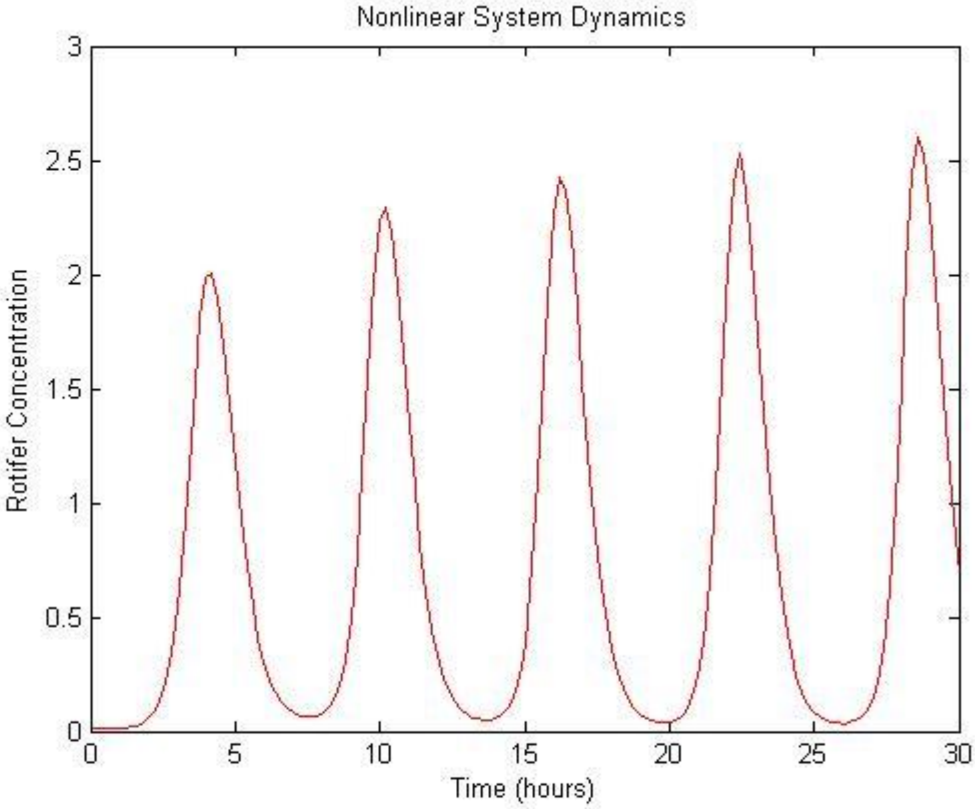
	nutrient	algae	rotifer
1	2	1	0.0100
2	2	1	0.0100
3	1.6938	1.4073	-0.0797
4	1.3492	1.7654	-0.0025
5	0.9852	2.0379	0.2279
6	0.6248	2.1961	0.5847
7	0.2931	2.2211	1.0300
8	0.0142	2.1063	1.5181

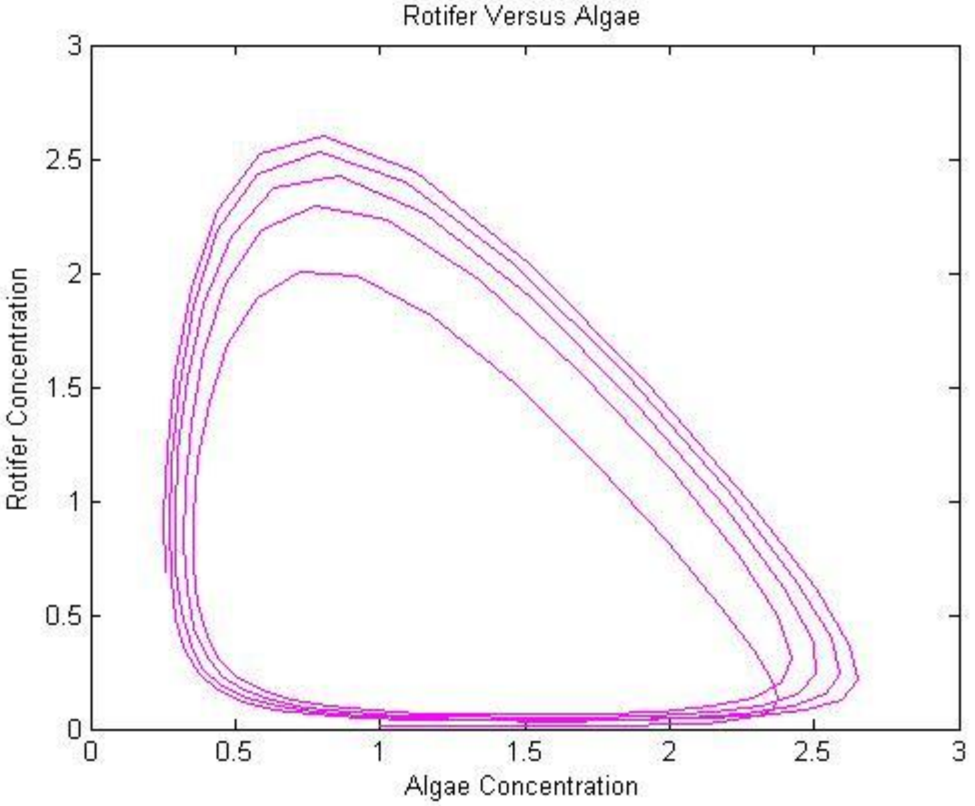
SECOND DATA

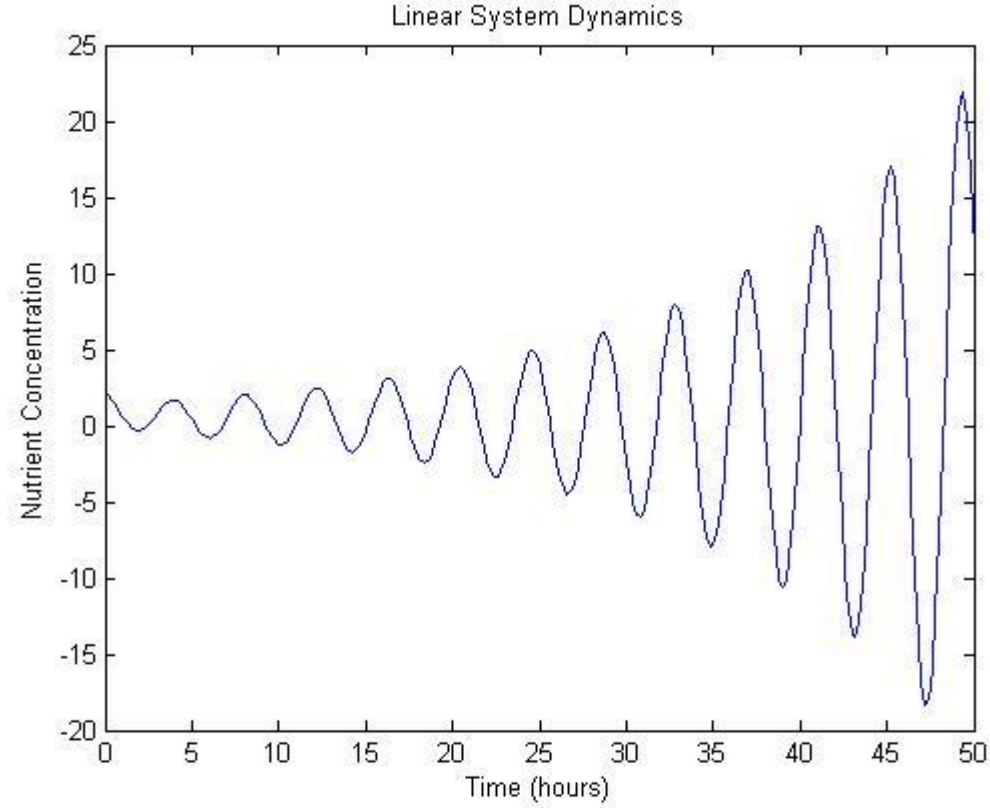
	nutrient	algae	rotifer
1	2	1	0.0100
2	0.7786	1.3099	0.0175
3	0.3394	1.4796	0.2245
4	0.1981	1.5589	0.5125
5	0.1789	1.5678	0.8250
6	0.2160	1.5178	1.1291
7	0.2819	1.4189	1.4026
8	0.3629	1.2822	1.6293

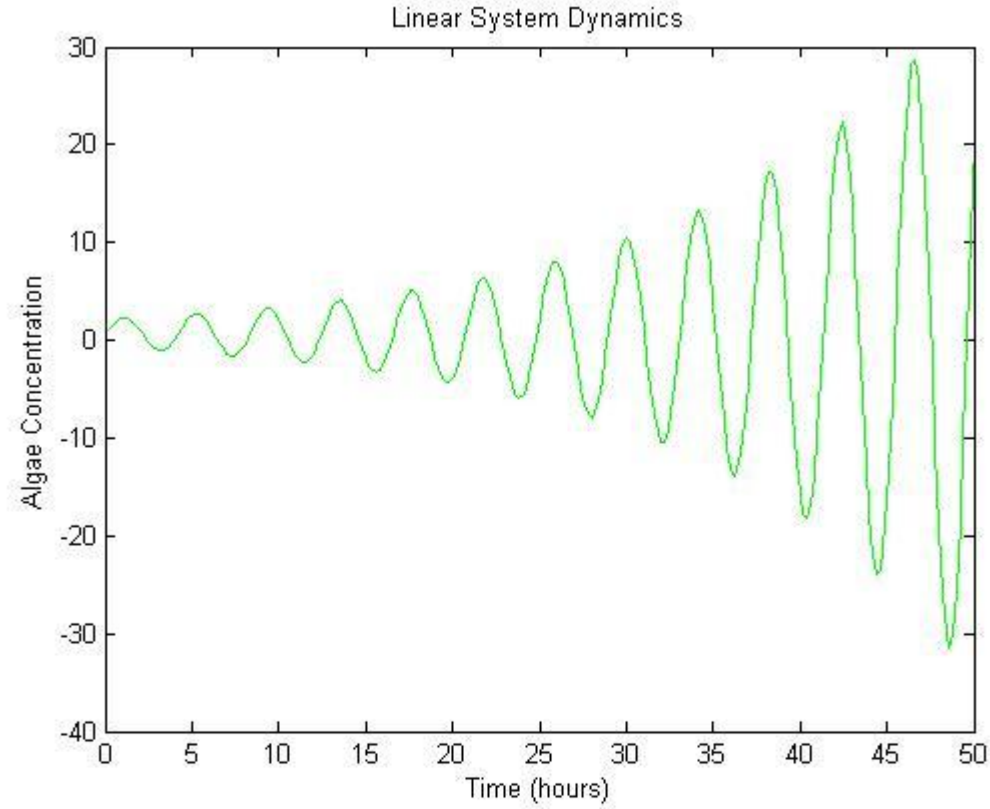

Nonlinear System Dynamics
Nutrient Concentration
Time (hours)
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0
5
10
15
20
25
30

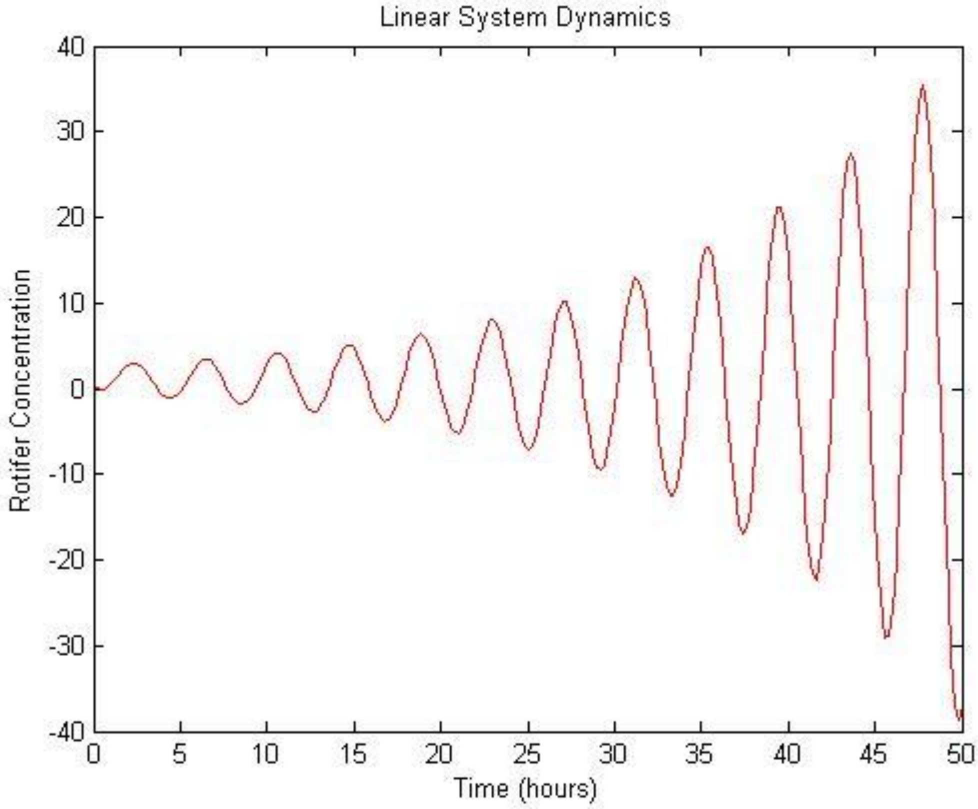

Nonlinear System Dynamics
Algae Concentration
Time (hours)
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30

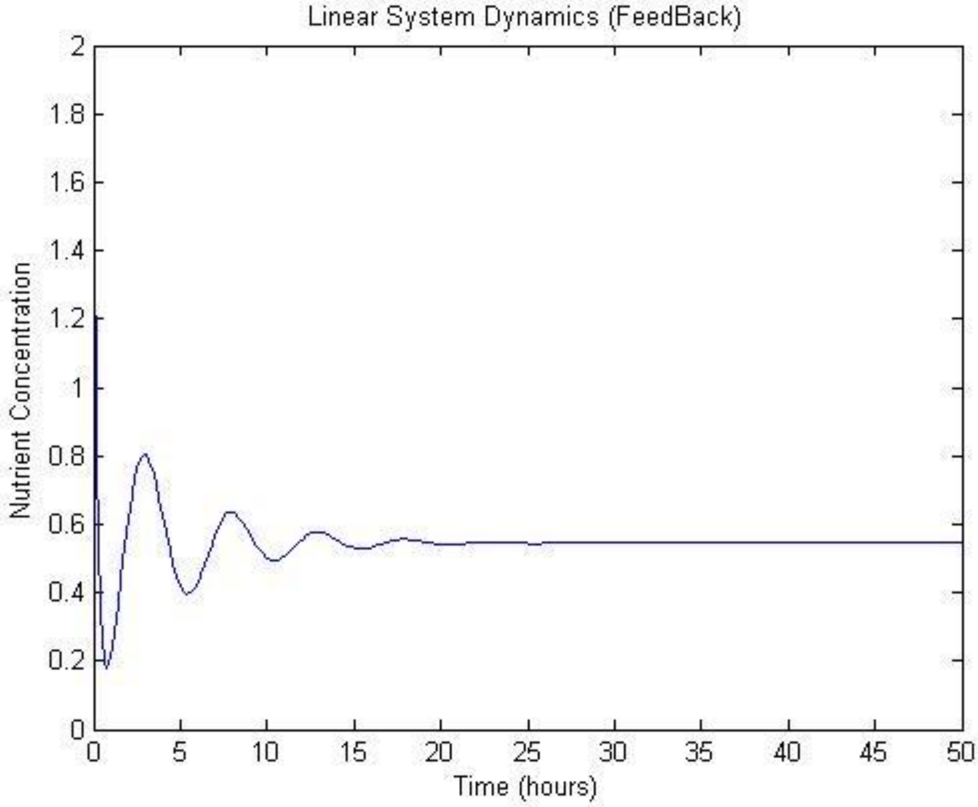

Nonlinear System Dynamics
Rotifer Concentration
Time (hours)
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30

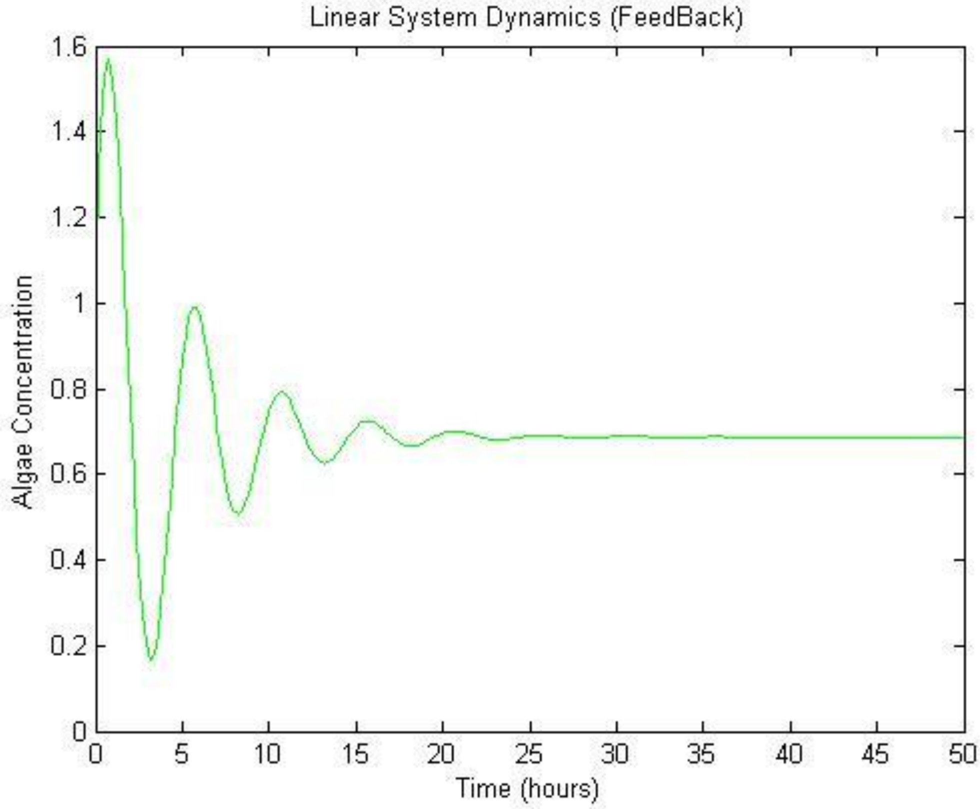

Rotifer Versus Algae
Rotifer Concentration
Algae Concentration
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2
2.5
3

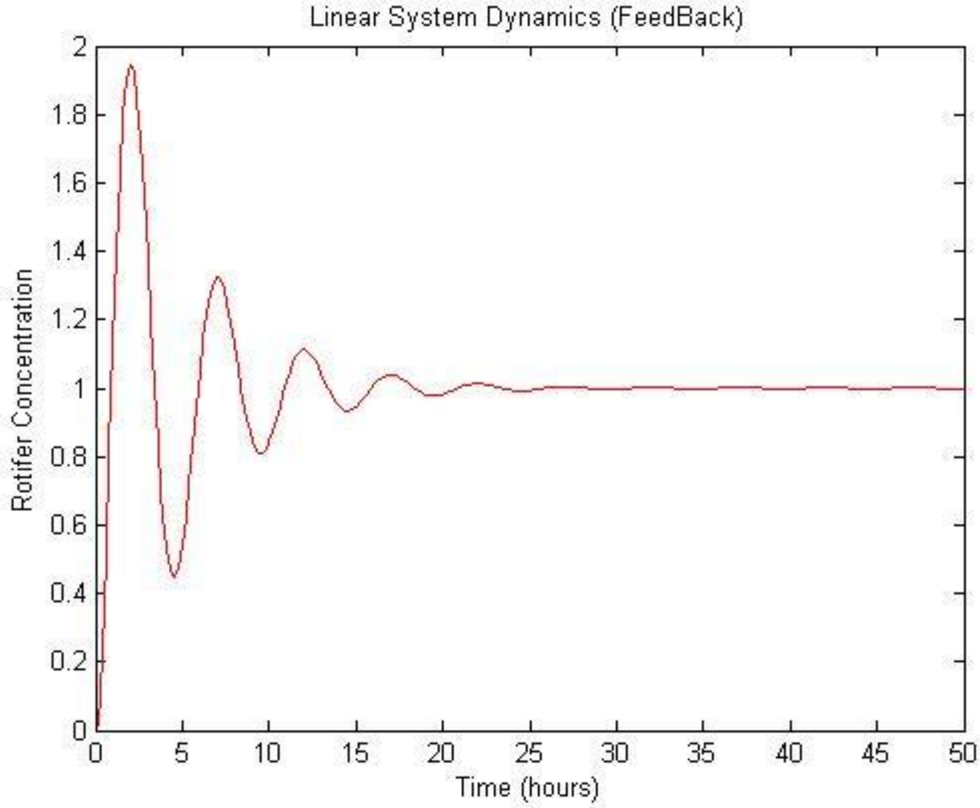

Linear System Dynamics
Nutrient Concentration
Time (hours)
25
20
15
10
5
0
-5
-10
-15
-20
0
5
10
15
20
25
30
35
40
45
50


Linear System Dynamics
Algae Concentration
Time (hours)
30
20
10
0
-10
-20
-30
-40
0
5
10
15
20
25
30
35
40
45
50


Linear System Dynamics
Rotifer Concentration
40
30
20
10
0
-10
-20
-30
-40
0
5
10
15
20
25
30
35
40
45
50
Time (hours)


Linear System Dynamics (FeedBack)
Nutrient Concentration
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35
40
45
50
Time (hours)


Linear System Dynamics (FeedBack)
Algae Concentration
Time (hours)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
5
10
15
20
25
30
35
40
45
50


Linear System Dynamics (FeedBack)
Rotifer Concentration
Time (hours)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
5
10
15
20
25
30
35
40
45
50

7. Kesimpulan

Dari beberapa permasalahan yang ada, yaitu pada rangkaian jaringan *nutrisi, algae dan rotifer* terdapat hubungan yang saling keterkaitan antara satu sama lain. Berdasarkan hasil analisis yang dilakukan diperoleh kesimpulan sebagai berikut:

1. Sistem yang diamati berupa sistem nonlinear.
2. Sistem nonlinear yang diperoleh dapat dilinearisasikan.
3. Sistem yang diperoleh dapat memiliki nilai aigen yang tidak imajiner.
4. Sistem teramati.
5. Sistem terkontrol.
6. Sistem tak stabil kemudian sistem diberikan *feed-beck* sehingga sistem dapat terstabilkan.

DAFTAR PUSTA

[1] G.F. Fussmann, S.P. Ellner, K.W. Shertzer, and N.G. Hairston. Crossing the hopf bifurcation in a live predator-prey system. *Science*, 290:1358–1361, 2000.

[2] M.G. Gore. *Spectrophotometry and Spectrofluorimetry*. Oxford University Press, 2000.

[3] T. Yoshida, L.E. Jones, S.P. Ellner, G.F. Fussmann, and N.G. Hairston. Rapid evolution drives ecological dynamics in a predator-prey system. *Nature*, 424:303–305, 2003.

[4] M. Chalfie, Y. Yu, G. Euskirchen, W.W. Ward, and D.C. Prasher. Green fluorescent protein as a marker for gene expression. *Science*, 263:802–805, 1994.

This writeup was done for an undergraduate controls course final project. The plant to be controlled is different, but the general format of the writeup is the same.

Controlling a Predator-Prey Chemostat

Sam Burden
Department of Electrical Engineering, University of Washington

Now a graduate student in controls at UC Berkeley!

***Abstract*—For this project, we designed several control systems for a predator-prey chemostat: one obtained using root locus methods and three which rely on full-state feedback, including one that utilizes the LQR optimal control technique. We also quantified the desired performance of the system and its robustness to disturbances and sensor noise. We applied all three controllers as well as the performance and robustness analysis to both a linear and a nonlinear model of the chemostat, and applied the observer-based controller to a nonlinear model which incorporates inaccuracies in measured parameters, observation delay, and limits on the rate of nutrient addition.**

I. Predator-Prey System

We designed and analyzed a variety of control systems for a predator-prey system in which the **chem**ical environment is **stat**ic, a *chemostat* (see Figure 1). Nutrients flow into the chemostat and mix with a soup of organisms; waste is removed to keep the level of fluid inside the container constant.

Our system contains both rotifers (the *predator*) and algae (the *prey*) with concentrations $b(t)$ and $a(t)$, respectively [1]. We distinguish the concentration of nutrients $n(t)$ and we control the rate at which nutrients are added to the system, $u(t)$.

To model the system, we assume the following:

1) Nutrients are digested at a rate proportional to the product of the concentrations of nutrients and algae.
2) Algae reproduce at a rate proportional to the rate of nutrient consumption, and they are eaten by rotifers at a rate proportional to the product of their concentrations.
3) Rotifers reproduce at a rate proportional to the rate at which they eat algae, and die when genetic mutations make the nutrients poisonous to them.
4) We can measure the concentration of rotifers in the system using a spectrofluorimeter [2].

This leads us to a set of governing differential equations of the form $\dot{x} = f(x, u)$:

$$\dot{x} = \begin{pmatrix} \dot{n} \\ \dot{a} \\ \dot{b} \end{pmatrix} = \begin{pmatrix} u - k_1 na \\ k_1 \alpha na - k_2 ab - k_3 a \\ k_2 \beta ab - k_4 nb \end{pmatrix}. \quad (1)$$

and the output equation $y = b(t)$. The (assumed constant) physical parameters are measured as $k_1 \approx 0.5$, $k_2 \approx 0.7$, $k_3 \approx 0.5$, $k_4 \approx 0.9$, $\alpha \approx 1.1$, $\beta \approx 2.0$.

We simulated this system using a constant nutrient addition rate $u(t) \equiv 1$. Figure 2 shows the concentration of nutrients, algae, and rotifer as a function of time.

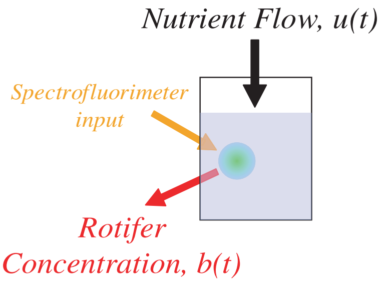


Fig. 1. System in which the **chem**ical environment is **stat**ic, a *chemostat*. Nutrients flow in at a rate $u(t)$, algae consume the nutrients and reproduce, rotifers consume the algae and die from consuming the nutrients. The rotifer concentration, $b(t)$, is measured with a *spectrofluorimeter*.

Note that all figures have captions!

The underlying syst ar when we note that the concentration of algae, $a(t)$, is roughly 180^o out of phase with the concentration of rotifer, as Figure 3 illustrates. This means that the maximum algae population approximately coincides with the minimum rotifer population, and the maximum rotifer with the minimum algae, one of the defining characteristics of a predator-prey system.

We would like to be able to control the concentrations of nutrients, algae, and rotifers to conduct various experiments, in particular experiments with constant concentrations and with $b \equiv 1$. Accordingly, we wish to design compensators to hold the system at a natural equilibrium point in the face of possible external “impulses”, including environmental effects like temperature variation and internal effects like evolution and deterioration of nutrients.

We begin by defining a set of desired performance specifications, then identify a system equilibrium point and linearize the system about that point. Then we create a battery of compensators for the system using root locus, full-state, and optimal control methods and apply them to both a linear and nonlinear model of the system using MATLAB’s *Simulink* toolbox. We analyze the effect of nutrient contamination and fluorescent deterioration on the performance of the root locus-based controller, and apply an observer-based controller to a more realistic model of the system which incorporates parameter uncertainty, observation delay, and limits on the rate of nutrient addition.

II. Performance Specification

As the rotifer population grows and threatens that of the algae, the algae respond by evolving to make themselves distasteful to their predators [3]. Accordingly, we wish to stabilize the concentration of rotifers at 1.0 to prevent the

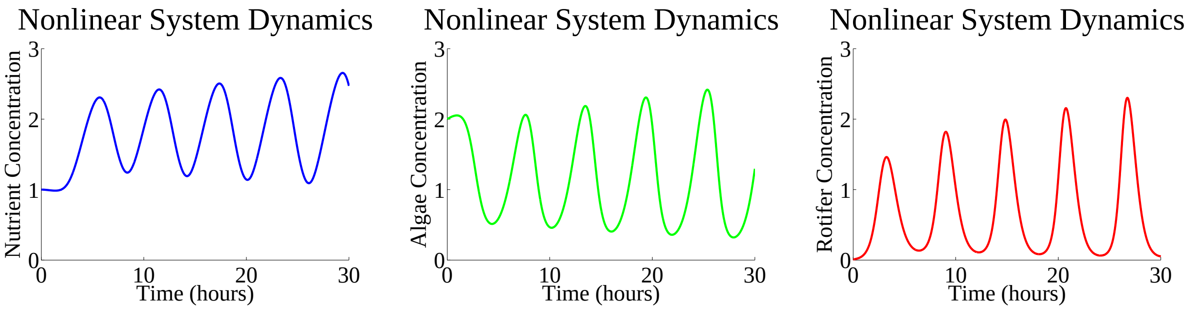


Fig. 2. Concentration of nutrients, algae, and rotifers versus time for the nonlinear system given by the equations in (1) with the initial condition n_0 = 2, a_0 = 1, b_0 = 0.01.

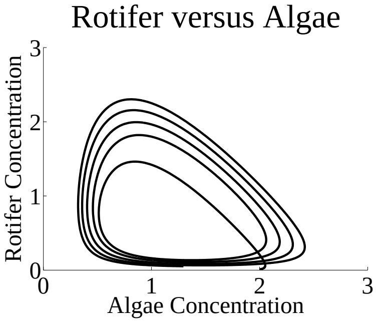


Fig. 3. Concentration of rotifer vs. concentration of algae over 30 hours. The two populations fluctuate out of phase, exhibiting one of the defining characteristics of a predator-prey system.

algae from evolving too swiftly. It takes, on average, ≈ 5 hours for a significant fraction of the algae to evolve. To keep evolution from invalidating our model of the system, we wish to stabilize the rotifer population in under 5 hours with the smallest overshoot possible overshoot. At the bare minimum, then,

$$P_O < 50\%, \; T_s < 5 \; hours. \tag{2}$$

If possible, however, we'd like to attain more stringent requirements, namely

$$P_O < \frac{1}{10}\%, \; T_s < 1 \; hour. \tag{3}$$

If we approximate the system as second-order, then we wish to place the dominant second-order poles at $p_{\pm} = -4 \pm 1.82j$.

III. Error Coordinates and Linearization

A. Equilibrium Point

We wish to control the behavior of this system near an equilibrium point which sets $b \equiv 1$. Solving the equations in (1) with $\dot{n} \equiv 0$, $\dot{a} \equiv 0$, and $\dot{b} \equiv 0$ near $b \equiv 1$ yields (besides the trivial solution at the origin) the desired equilibrium point,

$$x^* = \begin{pmatrix} n^* \\ a^* \\ b^* \\ u^* \end{pmatrix} = \begin{pmatrix} \frac{k_2+k_3}{\alpha k_1} \\ \frac{(k_2+k_3)k_4}{\alpha\beta k_1 k_2} \\ 1 \\ \frac{(k_2+k_3)^2 k_4}{\alpha^2 \beta k_2 k_2} \end{pmatrix}. \tag{4}$$

B. Linearization

If we think of (1) as having the form $\dot{x} = f(x, u)$, we can linearize the system around its equilibrium point by defining A, B, and C matrices

$$A = \frac{\partial f}{\partial x}\bigg|_{x=x^*}^{u=u^*}, \; B = \frac{\partial f}{\partial u}\bigg|_{x=x^*}^{u=u^*}, \; C = \frac{\partial y}{\partial x}\bigg|_{x=x^*}^{u=u^*} \tag{5}$$

and writing

$$\dot{x}_e = Ax_e + Bu_e, \quad y_e = Cx_e, \tag{6}$$

where $x_e = x - x^*$. In particular,

$$A = \begin{pmatrix} -\frac{(k_2+k_3)k_4}{\alpha\beta k_2} & -\frac{k_2+k_3}{\alpha} & 0 \\ \frac{(k_2+k_3)k_4}{\beta k_2} & 0 & -\frac{(k_2+k_3)k_4}{\alpha\beta k_1} \\ -k_4 & \beta k_2 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; C = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}.$$

C. Natural Response

If we substitute the measured system parameters for the variables in A, its eigenvalues are

$$\lambda = -0.830, \quad 0.0643 \pm 1.52j. \tag{7}$$

Since two of the poles lie in the right half-plane, our desired equilibrium point is *unstable*, meaning that unless we manage to set the system's state precisely at that point, it will not remain near there. Therefore, we must control the system to keep it near the desired state.

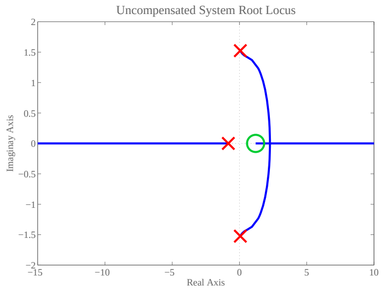


Fig. 4. Root locus of the chemostat plant given in (8).

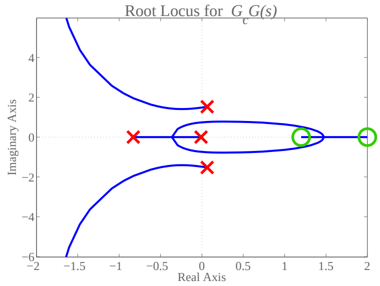


Fig. 5. Root locus of the compensated chemostat system.

D. *Transfer Function*

We can obtain a transfer function for the plant system from the matrices in (5) using the equation $G(s) = C(sI - A)^{-1}B$ and the measured system parameters:

$$G(s) = \frac{-0.9s + 1.08}{s^3 + 0.701s^2 + 2.22s + 1.93}. \qquad (8)$$

IV. Root Locus Design

We'll create our first controller for the chemostat system using root locus design. The open-loop root locus of the system is shown in Figure 4.

Unfortunately, root locus methods turn out to be particularly unsuitable for this particular system. Most of the usual tricks—most notably PID control—don't even make the system stable, let alone yield the desired performance. We found that the configuration shown in Figure 5 was one of the simplest that seemed to reliably stabilize the system. The compensator has transfer function

$$G_c(s) = \frac{-0.5s + 1}{100s + 1} \qquad (9)$$

with a gain of 75. However, this configuration does not even remotely achieve our desired performance specification: the impulse response over 50 hours is shown in Figure 6. The linear and nonlinear systems perform qualitatively similar, having roughly the same percent overshoot and settling time.

The problem with applying standard root locus techniques to this system is that the system has a zero in the right half-plane; this means the plant's complex-conjugate poles are typically drawn into the right-half plane as the closed-loop gain is increased. If one adds enough zeros in the left-half plane to counteract this effect, the system order is implicitly increased (you create 'poles at infinity') and the system is still unstable.

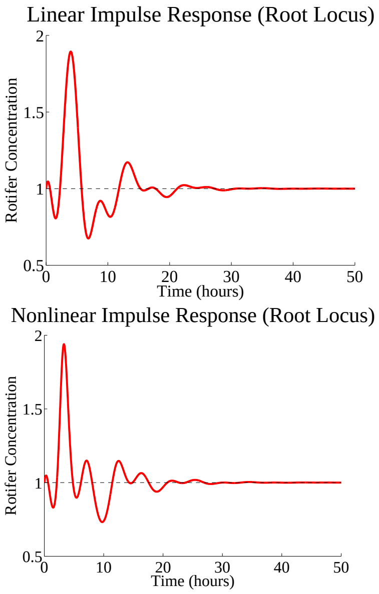


Fig. 6. Impulse response of the linear and nonlinear models with the root locus-based compensator given in (9).

One possible solution to this problem would be to change the chemostat system itself. Since the zeros of the plant are determined solely by the system's input-output relationship, we experimented with the transfer function obtained by altering the B and C matrices. The easiest modification to realize physically would be to add more sensors, so we focused on the C matrix. The simplest resulting transfer function is obtained when we measure the algae concentration instead of the rotifer concentration. One can imagine achieving this in the physical system by genetically altering the algae to produce GFP instead of the rotifers, then using the same measurement technique as before.

This modification yields the transfer function

$$G_a(s) = \frac{0.771s + 0.884}{s^3 + 0.701s^2 + 2.22s + 1.93}. \qquad (10)$$

In particular, the zero has now been moved to the left half-plane, where our usual root locus techniques apply quite well. We designed a physically-realizable lead compensator to control this system,

$$G_{a,c}(s) = \frac{1.3s + 1}{\frac{1}{10}s + 1} \qquad (11)$$

with a gain of 3.3. The linear and nonlinear system's impulse responses are given in Figure 8. Our hastily-designed controller yields the desired settling time, though the percent overshoot still needs some work.

V. Disturbances and Sensor Problems

A. *Disturbances*

The chemostat system is largely safe from outside disturbances. However, the nutrient batch is susceptible to contamination. As a consequence, we wish to analyze the effect a

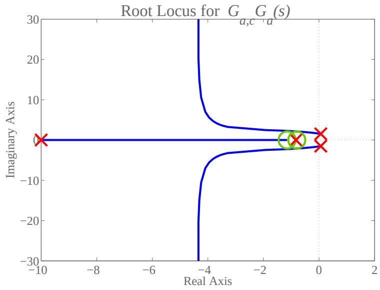


Fig. 7. Root locus of the compensated modified chemostat system (outputs algae concentration instead of rotifer concentration).

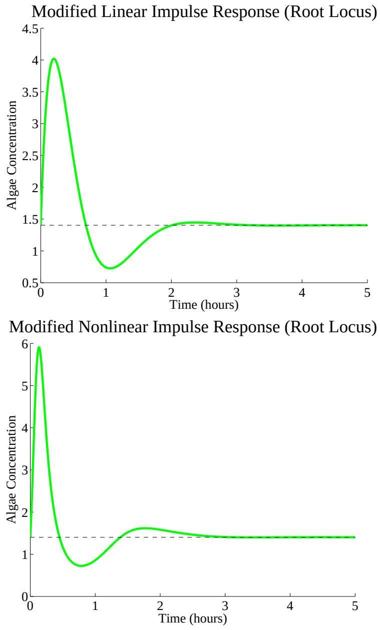


Fig. 8. Impulse response of the modified linear and nonlinear models with the root locus-based compensator given in (11).

bad batch of nutrients would have on our controlled system, to know whether we will have to interrupt an experiment if contamination occurs.

When the nutrients become contaminated, the effective addition to nutrient concentration from added nutrients decreases, algae which consume them reproduce more slowly, and rotifers which consume them are less likely to be poisoned. Then if we assume that the algae will consume contaminated nutrients at the same rate as they would for uncontaminated, nutrient contamination can be modeled as a proportional decrease in the rate of nutrient addition:

$$u(t) \mapsto \delta u(t), 0 < \delta \leq 1.$$

This modification affects our controlled system in two ways: first, it scales down the equilibrium rate of nutrient addition, $u^* \mapsto \delta u^*$; second, it scales down our error-coordinate control signal, $u_e(t) \mapsto \delta u_e(t)$ (see Figure 9). A typical response for the linear and nonlinear systems with $u_e(t) \equiv 0$ and $\delta = 0.5$ is shown in Figure 10.

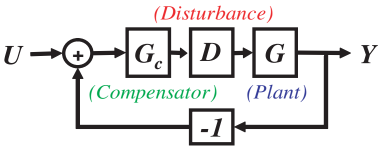


Fig. 9. Block diagram of compensated system with disturbance.

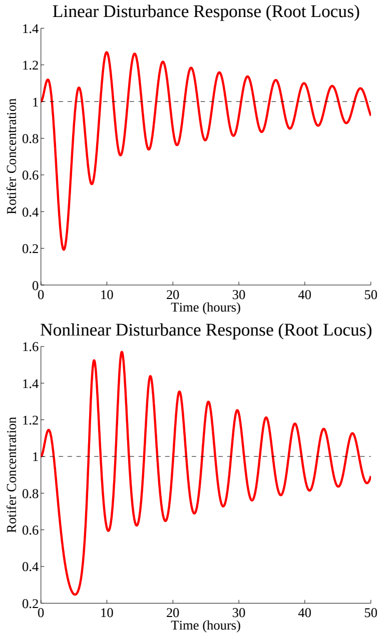


Fig. 10. Disturbance response of the linear and nonlinear models (root locus-based compensator).

Since our root locus-based controller has such poor performance, the effect of the disturbance takes a long time (more than 50 hours!) to damp out.

If we assume that $u_e(t)$ is small compared to u^*, then we can approximate this disturbance as a step input added between our compensator and the system model (see Figure 11). The advantage to this method is that the closed-loop transfer function is easily found by setting $U(s) \equiv 0$ and noting that

$$Y(s) = G(s)(D(s) - G_c(s)Y(s)),$$

so

$$T(s) = \frac{Y(s)}{D(s)} = \frac{G(s)}{1 + G_c(s)G(s)}.$$

Using $G(s)$ from (8) and $G_c(s)$ from (9) and our corresponding choice for the gain we find (numerically) that $T(s)$ is stable. Now using the disturbance $D(s) = \delta/s$, the final-value theorem yields $y(\infty) = 0.0130\delta$, so the disturbance induces a (relatively small) steady-state error.

B. Sensor Problems

We measure the concentration of rotifers using a spectrofluorometer. This device excites a *green fluorescence protein* (GFP) fluorophor [4] we've engineered the rotifers

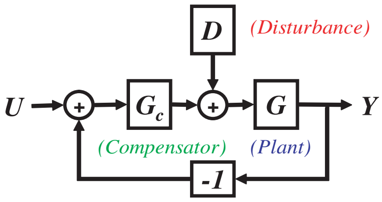


Fig. 11. Block diagram of compensated system with step disturbance, an approximation to the disturbance model in Figure 9.

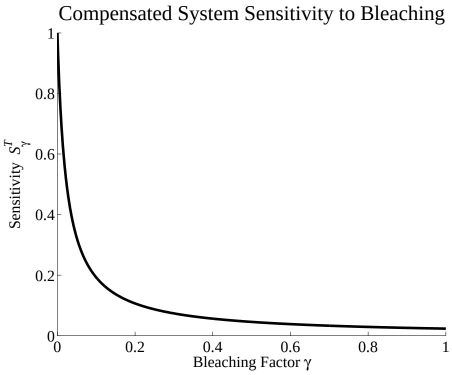


Fig. 12. Sensitivity of root locus-controlled system with respect to fluorophor bleaching ($s = \sigma + j\omega = 0$).

to produce. Over time, the continual activity *bleaches* the GFP, resulting in a smaller output than we would expect for a given concentration, $(y = b) \mapsto (y = \gamma b)$, $0 < \gamma \leq 1$.

We can determine the *sensitivity* of the compensated system to γ by defining the sensitivity of our system S_γ^T as

$$S_\gamma^T = \frac{\partial T}{\partial \gamma}\frac{\gamma}{T}. \qquad (12)$$

Now, since $T(s) = (G_c(s)G(s))/(1 + G_c(s)G(s))$, we can compute this sensitivity quite easily with the aid of a computer. Using software that works with symbolic expressions, we found the exact relationship between the sensitivity of the system and γ. With $s = \sigma + j\omega = 0$ (the sensitivity is always nearly 1 at moderate-to-high frequencies),

$$S_\gamma^T(\gamma) = \frac{1.93}{1.93 + 81\gamma}. \qquad (13)$$

A plot of the sensitivity for $\gamma \in (0, 1]$ is shown in Figure 12. As shown there, the sensitivity of the system to γ is fairly low when $\gamma > 0.2$.

VI. FULL-STATE FEEDBAC

This should have been written out for this system and the analysis of the matrix explained.

The *controllability matrix* given by

$$M = (B\ AB\ A^2B)$$

has rank 3, so the system is controllable, can use full-state feedback to control the system.

Our performance specification from Section II aims to place the chemostat system's dominant second-order poles at $p_\pm = -4 \pm 1.82j$. The system is third-order, so we place the third pole at -20. This is more than five times as negative as the dominant poles, so a second-order approximation should be valid. Using the gain matrix $K = (K_1\ K_2\ K_3)$, we want to solve

$$(s + 20)(s + 4 + 1.86j)(s + 4 - 1.86j) = |sI - (A - BK)|$$

for the three gains. Since the equation is linear in the gains, the (unique) solution is readily found: $K = (27.3\ 309\ 68.1)$. This gain matrix is connected to the system in the manner illustrated in Figure 13.

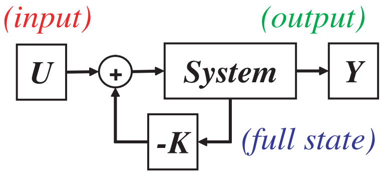


Fig. 13. Block diagram illustrating how a gain matrix is connected to a system when using full-state feedback.

Figure 14 shows the responses of the linear and nonlinear systems with feedback control to an impulse; the two systems respond nearly identically.

VII. OBSERVER DE

This should have been written out for this system and the analysis of the matrix explained.

The *observability matrix* defined by

$$O = \begin{pmatrix} C \\ CA \\ CA^2 \end{pmatrix}$$

has rank 3, so the system is observable, which means we can design an observer and use full-state feedback to control the system.

We want the observer's settling time to be much less than the chemostat's, so we need to place the observer's poles far to the left of the plant's dominant poles. We found that if the observer poles are too negative, the observer actually approximates the nonlinear system quite poorly, because its dynamics are too fast; instead, we choose the place the poles at -6 on the real axis. Though this is less than ten times as negative as the plant's dominant poles, it yields good performance. Using the gain matrix $L = (L_1\ L_2\ L_3)^T$, we want to solve

$$(s + 6)^3 = |sI - (A - LC)|$$

for the gains. Once again the equations are linear, and the n is readily found: $L = (29.1\ 85.6\ 17.3)^T$. This atrix is incorporated into an *observer* (see Figure 15) nnected to the system in the manner illustrated in 16.

re 17 shows the responses of the linear and nonlinear s with observer-based feedback control to an impulse. The two systems respond similarly, though their trajectories do not match so well as with ideal full-state feedback.

To test how well our observer can track the nonlinear system far away from equilibrium, we also wanted to simulate

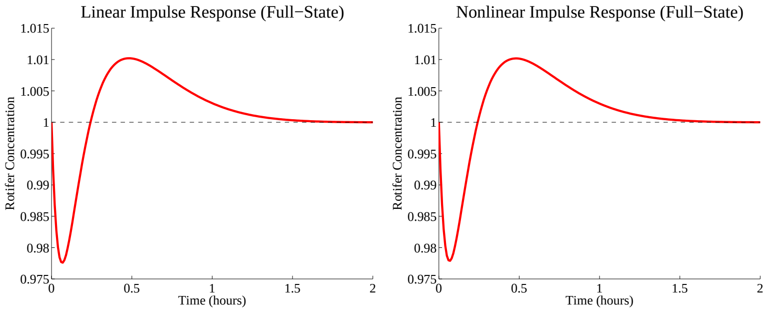


Fig. 14. Responses of the linear and nonlinear system with full-state feedback to an impulse; the systems respond nearly identically to the impulse.

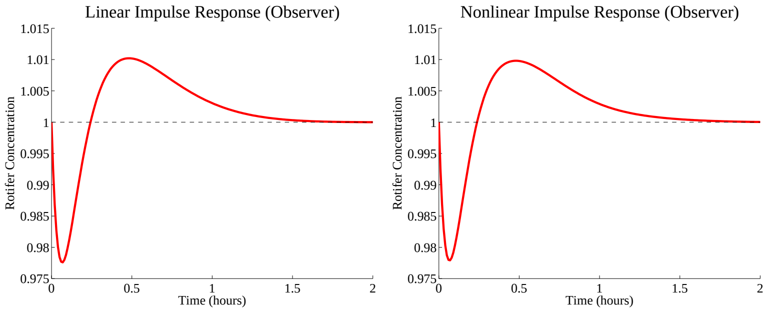


Fig. 17. Responses of the linear and nonlinear system with observer-based full-state feedback to an impulse; the two systems respond similarly to the impulse.

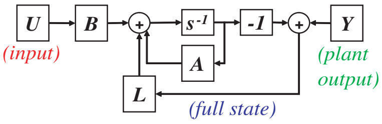


Fig. 15. Block diagram illustrating the construction of the observer used in Section VII.

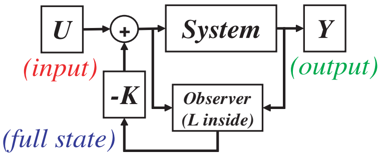


Fig. 16. Block diagram illustrating how an observer is connected to a system using full-state feedback.

the system starting with $n(0) = 0$ and all other variables at their equilibrium values. Figure 18 shows the nonlinear system's response in all three state variables along with the observer's prediction from this initial condition. The observer manages to track each state variable almost exactly over the whole simulation.

VIII. Optimal Control

The nutrients we use to fuel the chemostat system are expensive, so we also wanted to design a controller that uses the least amount of nutrients possible while still achieving our performance requirements. To achieve this optimality, we first relax our performance requirements to $P_O \approx 30\%, T_s < 5\ hours$. Then we use *Linear-Quadratic Regulation*, or LQR, to choose gains for a full-state feedback controller that are optimal in some sense. With LQR, we minimize a cost functional of the form

$$J = \int_0^\infty x(t)^T Q x(t) + u(t)^T R u(t)\ dt, \tag{16}$$

where Q is an $(n \times n)$ (constant) matrix and R is a (constant) scalar. Intuitively, a more heavily-weighted Q translates to a decreased settling time for the system, while a large R corresponds to a smaller control input.

Since we are interested in controlling the rotifer concentration and minimizing the control input, we weight these quantities more heavily in Q_0 and R_0, our initial choice for the cost parameters:

$$Q_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix}, \ R_0 = 5. \tag{17}$$

With the base values for Q and R chosen, we find (roughtly) the smallest value of ξ for which the choice $Q = \xi Q_0$, $R = (1-\xi)R_0$ yields acceptable system performance. Figure 19 shows the linear system's impulse response for a range of ξ.

Based on the responses shown in the figure, we select $\xi = 0.8$, which yields the gain matrix $K = (1.28\ 0.285\ \ -1.10)$; this matrix is connected to the system in the manner shown in Figure 13. This choice gives us 30% overshoot and a settling time of roughly 5 hours, which matches the performance bounds we gave above. Since we ultimately wish to minimize the amount of nutrients we use, it makes sense to meet the upper bound of our performance specification.

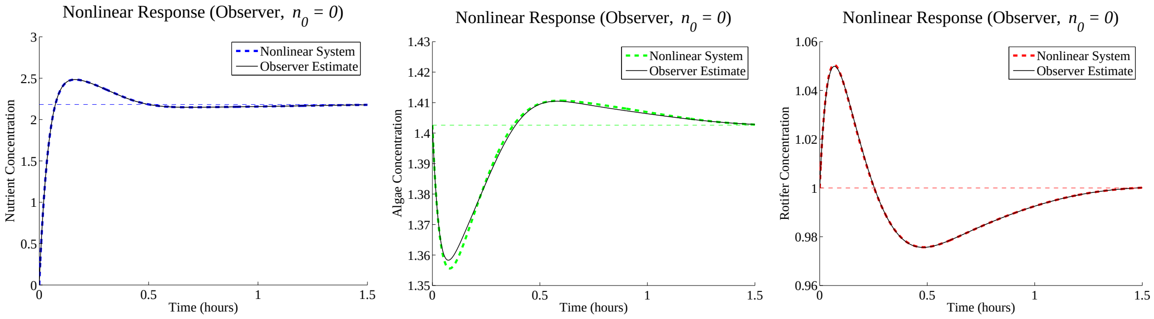
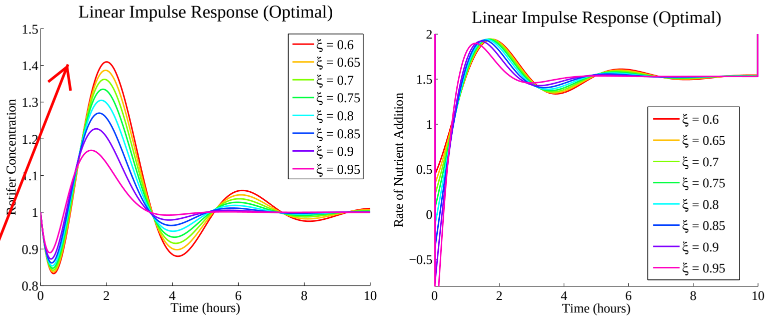


Fig. 18. Nonlinear system response starting from $n(0) = 0$, other state variables at their equilibrium values, with estimated state from the (linear system-based) observer.



Combination of multiple plots in one is very cool, saves space, and allows for easy comparisons

on and nutrient addition rate impulse response for the LQR-controlled linear system over a range of ξ; $Q = \xi Q_0$, $R = (1-\xi)R_0$.

Figure 20 shows the impulse responses of the linear and nonlinear systems, as well as the cumulative nutrient addition beyond the contribution from the equilibrium rate u^*. The linear and nonlinear systems respond quite similarly to the impulse, which is in agreement with the result we obtained with our original ideal full-state feedback controller (see Figure 14). That the optimal control scheme stabilizes the system using less nutrients than would have been used by the system at equilibrium suggests that we made a good choice for our cost function.

Our full-state feedback controller uses $\approx$ 400 units of nutrients to stabilize the system; though it meets much more stringent performance requirements, the amount of nutrient saved with the optimized controller is startling. Our root locus compensator, which has far worse performance than our optimized controller, uses $\approx$ -4 units of nutrients to stabilize the system, which is comparable to the optimized system.

IX. A More Realistic Model

The model given in (1) is, at best, a rough approximation to the actual system's behavior. To be more thorough in our evaluation of a given controller, therefore, we ought to check its performance when applied to a more realistic model. Specifically, we add the following assumptions to the list in Section I:

5) The uncertainty in the measurement of the parameters k_1, α, and β is 10%.
6) There could be up to 10 minutes of observation delay ($y(t) = b(t-\tau)$, $0 \leq \tau \leq \frac{1}{6}$).
7) We cannot actively remove nutrients from the system, and we cannot add them faster than 4 ($0.0 \leq u \leq 4.0$).

Since it provided the best performance without having direct access to the system's full state, we test the observer-based controller's robustness by applying it to the nonlinear model of our system obtained by implementing these additional assumptions. Using precisely the same controller we presented in Section VII, we were unable to implement the additional assumptions listed above. Instead, we designed a full-state feedback controller to meet relaxed performance specifications ($P_0 < 1\%$, $T_s < 5$ $hours$) and slowed the response of the observer to make it less sensitive to unexpected behavior from the nonlinear model. The resulting gain matrices are $K = (12.9\ 6.03\ \ -14.6)$, $L = (-0.23\ 26.9\ 11.3)^T$.

We first consider qualitatively the effect of each of the assumptions separately, then present the result of their simultaneous application. Parameter uncertainty has a small effect on percent overshoot and settling time, but increases the steady-state error considerably. Limiting the rate of nutrient addition primarily increases settling time. Adding observation delay increases the settling time, percent overshoot, and steady-state error. Figure 21 illustrates the effect of all three model enhancements applied simultaneously, $\tau = 10$ minutes.

X. Discussion

In this project, we designed a variety of controllers for a chemostat system, applied them to both a linear and a nonlinear model of the system, and evaluated their perfor-

A discussion is crucial! What worked? What didn't? What would you do next?

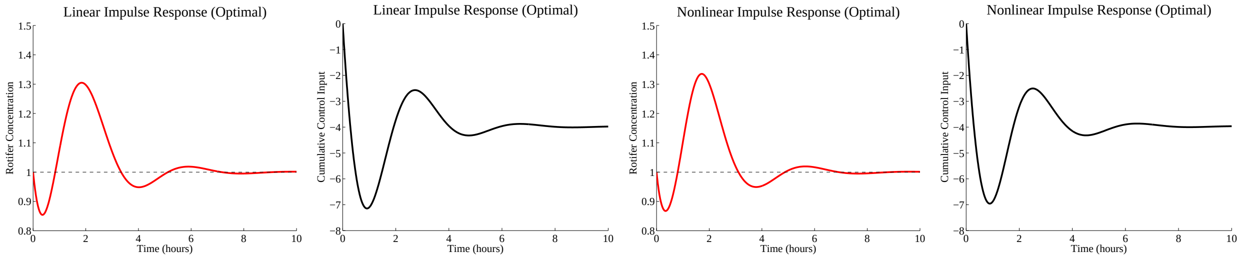


Fig. 20. Rotifer concentration and cumulative control input (nutrient addition beyond the anticipated equilibrium rate u^*) for the linear and nonlinear systems.

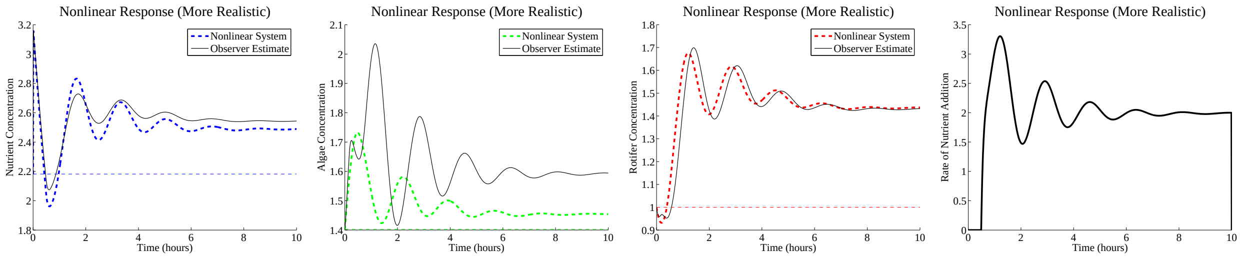


Fig. 21. Impulse response of the more realistic nonlinear model with observer-based full-state feedback control. Parameter uncertainty is 10%, observation delay is 10 minutes.

mance. We were able to design controllers which utilize the system's full state, either directly or through an observer, to meet a fairly rigorous performance specification for the system's impulse response near equilibrium. We were also able to optimize the rate of nutrient addition subject to given performance constraints.

However, the controller we designed without access to the system's full state (the root locus-based controller) fell short of the desired performance. But, we showed that if we redesigned the chemostat so that we could measure the concentration of algae instead of the concentration of rotifers, we could meet the performance specification using a root locus-based compensator.

We also showed that by relaxing the performance specifications, we were able to stabilize a more realistic model of the chemostat system using an observer-based full-state feedback controller, though we weren't quite able to meet our desired performance specifications in the case that there is a ten-minute observation delay and large ($\approx 10\%$) parameter uncertainty.

Further refinements on the work presented here ought to include more rigorous disturbance and sensitivity analysis, an investigation into the range of validity for the linear model, and application of our controllers to an even more realistic model of the system which models algae evolution and devolution in response to changes in the rotifer concentration.

References

[1] G.F. Fussmann, S.P. Ellner, K.W. Shertzer, and N.G. Hairston. Crossing the hopf bifurcation in a live predator-prey system. *Science*, 290:1358–1361, 2000.

[2] M.G. Gore. *Spectrophotometry and Spectrofluorimetry*. Oxford University Press, 2000.

[3] T. Yoshida, L.E. Jones, S.P. Ellner, G.F. Fussmann, and N.G. Hairston. Rapid evolution drives ecological dynamics in a predator-prey system. *Nature*, 424:303–305, 2003.

[4] M. Chalfie, Y. Yu, G. Euskirchen, W.W. Ward, and D.C. Prasher. Green fluorescent protein as a marker for gene expression. *Science*, 263:802–805, 1994.