Lab 10

Fourier Transformation and Finite Difference Model

Handout – print version of Lecture on *Marine Modelling* April 3, 2009

Katja Fennel, Oceanography, Dalhousie University

10.1

1 Outline

Outline

Plan for today:

- FFT example (script: `FFT_example.m`)
- Finitie Difference Approximation of N and P in a bottle (script: `NP_bottle.m`)

10.2

2 FFT Example

fft example objective

Building a test case:

- Build a periodic signal from two waves of different frequencies (add noise too).
- Pretend we don't know the frequencies, perform Fourier Transformation in order to recover those frequencies.
- Compare recovered frequencies with the known frequencies we chose to create our signal to begin with.

10.3

fft example

```
% 1. Create an artificial data set

deltat = .001; % time step
t = 0:deltat:1.023; % time vector
n = length(t); % we have 1024 data points
f1 = 50; f2 = 120;  % two frequencies: f1 and f2

% our artificial time series:
% two sine waves and random noise
y = sin(2*pi*f1*t)+2*sin(2*pi*f2*t)+0.5*randn(size(t));

% 2. calculate FFT and power spectrum
Y = fft(y);
Py = Y.*conj(Y); % |Y|^2
```

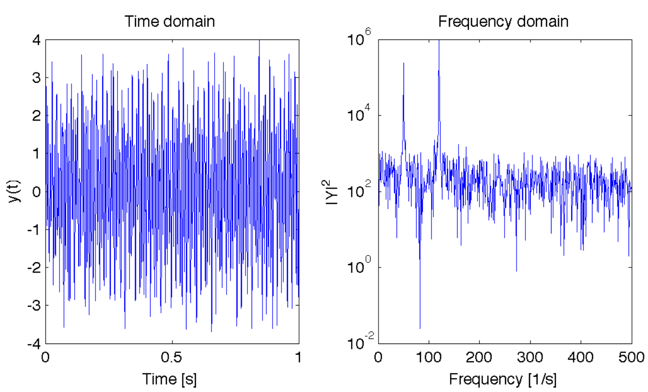
10.4

```
% 3. plot signal and power spectrum

Fs = 1/deltat; % sampling frequency

f = [0:n/2-1]/n*Fs; % frequency intervals
% for plotting; only up to the Nyquist
% frequency: 1/(2*deltat) = Fs/2

Py(n/2+1:n)=[]; % chop off Fourier coeffs
% at and above Nyquist frequency
```

10.5

```
figure
subplot(1,2,1)
plot(t,y)
set(gca,'FontSize',12)
xlabel('Time [s]')
ylabel('y(t)')
title('Time domain')
axis([0 1 -4 4])

subplot(1,2,2)
semilogy(f,Py)
set(gca,'FontSize',12)
xlabel('Frequency [1/s]')
ylabel('|Y|^2')
title('Frequency domain')
axis([0 500 10^(-6) 10^(-1)])
```

10.6



10.7

For checking out the size and frequencies of the peaks one can use Matlab's `sort` command

```
>> [Peaks IFreqs] = sort(-Py);
>> abs(Peaks(1:5))
ans =
   1.0e+06 *
    1.0347   0.2416   0.0255   0.0158   0.0120   0.0062   0.0049

>> f(IFreqs(1:5))
ans =
120.1172   49.8047   50.7813   119.1406   121.0938
```

10.8

Note that Matlab sorts in ascending order by default (it starts with the smallest value). Since we are looking for the largest peaks (largest values in `Py`) we want to order starting with the largest value. We can use the simple trick of ordering `-Py`.

Quantities involved in FFT

`y`	data
`n = length(y)`	number of samples
`dt`	time increment
`Fs = 1/dt`	Sampling rate
`t = [0:n-1]/Fs`	total time vector
`Y = fft(y)`	Fourier transform
`abs(Y)`	magnitude of Fourier coefficients
`Y.*conj(Y)`	power
`f=[0:n-1]/n*Fs`	frequency; cycles per time unit
`Fs/2 = 1/2/dt`	Nyquist frequency
`p = 1./f`	period; unit time per cycle

Note: You only need to look at the first half of the Fourier coefficients because the second half is a reflection about the Nyquist frequency. 10.9

3 N and P in a bottle

N and P in a bottle

Phytoplankton culture, P [mmol N/m^3], in a bottle with nutrient, N [mmol N/m^3], and you know that uptake occurs according to Michaelis-Menten kinetics; you also know the uptake parameters approximately.

$$\begin{aligned}\frac{dP}{dt} &= \mu_{max}\frac{N}{k_N+N}P - rP \\ \frac{dN}{dt} &= -\mu_{max}\frac{N}{k_N+N}P + rP\end{aligned}$$

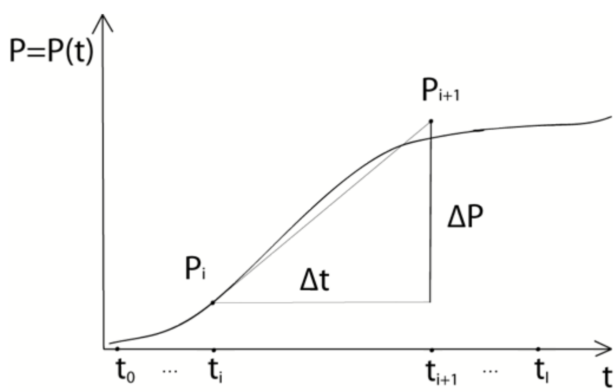
Recipe: replace $\frac{dP}{dt}$ by $\frac{\Delta P}{\Delta t}$ and $\frac{dN}{dt}$ by $\frac{\Delta N}{\Delta t}$ 10.10

Conventions:

We want to solve for discrete time steps, t_i between t_0 and t_{end}:

$$t_i = t_0 + i \times \Delta t \quad (i = 0, \cdots, I)$$

Refer to N, P at t_i as N_i, P_i.



10.11

"Euler forward"

$$
\begin{aligned}
\frac{P_{i+1}-P_i}{\Delta t} &= \mu_{max}\frac{N_i}{k_N+N_i}P_i - rP_i \\
\frac{N_{i+1}-N_i}{\Delta t} &= -\mu_{max}\frac{N_i}{k_N+N_i}P_i + rP_i
\end{aligned}
$$

and rearranging yields:

$$
\begin{aligned}
P_{i+1} &= P_i + \Delta t\; (\mu_{max}\frac{N_i}{k_N+N_i}P_i - rP_i) \\
N_{i+1} &= N_i + \Delta t\; (-\mu_{max}\frac{N_i}{k_N+N_i}P_i + rP_i)
\end{aligned}
$$

10.12

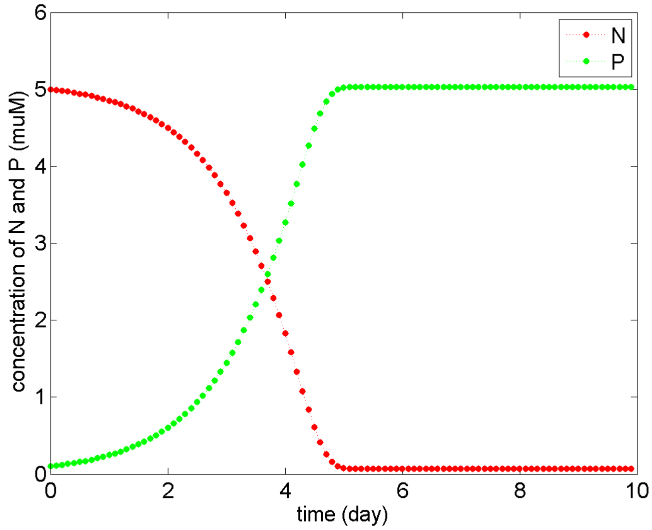
N and P in a bottle

```
clear % clear the workspace before we do anything new

% 1.) set constants
del_t  = 0.1;   % in days
k_N    = 0.75;  % in muM
mu_max = 1.2;   % in days-1
r      = 0.1;   % in days-1
n_max  = 100;    % maximum number of timesteps

% 2.) initialize state variables N and P
% (and time -- only for plotting purposes)
N(1) = 5.0;  % in muM; N at t0
P(1) = 0.1;   % in muM; P at t0
t(1) = 0;     % in days; t0
```

10.13

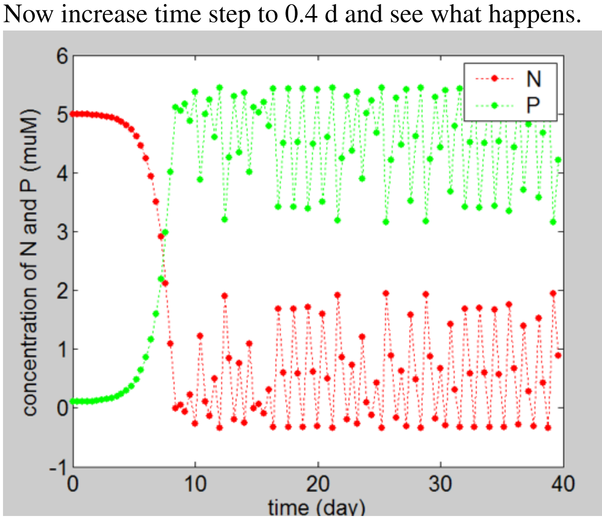
```
% 3.) calculate numerical solution from t0 to t_end
for n=2:n_max
    t(n) = t(n-1)+del_t;
    uptake = mu_max*N(n-1)/(k_N+N(n-1));
    P(n) = P(n-1) + del_t*( uptake - r)*P(n-1);
    N(n) = N(n-1) + del_t*(-uptake + r)*P(n-1);
end

% 4.) plot solution
hp=plot(t,N,'r.:',t,P,'g.:');
set(hp,'MarkerSize',16)
set(gca,'FontSize',16)
xlabel('time (day)')
ylabel('concentration of N and P (muM)')
legend('N','P')
```

10.14



So far we have used a time step of 0.1 d. 10.15

Now increase time step to 0.4 d and see what happens. 10.16



10.17

$$P_{i+1} = P_i + \Delta t \, (\mu_{max} \frac{N_i}{k_N + N_i} P_i - r P_i)$$

$$N_{i+1} = N_i + \Delta t \, (-\mu_{max} \frac{N_i}{k_N + N_i} P_i + r P_i)$$

Note that only concentrations from "previous" time point n are used to arrive at "next" time point (n+1).

$$C^{n+1} = f(C^n) \quad \textbf{"explicit scheme"}$$

Think about the implications in terms of "tangent on the curve" or "control volume".

Wouldn't it be more "accurate" to allow C to change over the time period Δt?

$$C^{n+1} = f(C^n, C^{n+1}) \quad \textbf{"implicit scheme"}$$

10.18

$$C^{n+1} = f(C^n, C^{n+1}) \quad \textbf{"implicit scheme"}$$

May seem tricky, but is tractable.

Leads to a set of simultaneous equations that need to be solved for each time step (not bad in one dimension, but gets expensive for higher spatial dimensions).

Example for nutrient uptake:

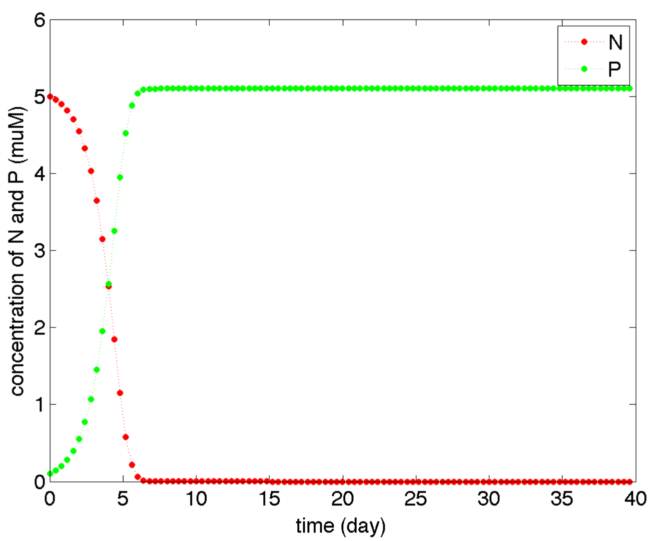
$$\frac{dN}{dt} = -\mu \frac{N}{k+N} P$$
$$\frac{N^{n+1} - N^n}{\Delta t} = -\mu \frac{N^{n+1}}{k+N^n} P^n$$

10.19

$$N^{n+1} = \frac{N^n}{1 + \frac{\mu \Delta t P^n}{k+N^n}}$$

A non-negative number is divided by a positive number. This scheme is *positive definite*.

Implement this for our example! 10.20



10.21

Key change to make the scheme implicit:

```
% calculate numerical solution from t0 to t_end
for n=2:n_max
    t(n) = t(n-1)+del_t;
    cff = mu_max*del_t*P(n-1)/(k_N+N(n-1));
    N(n) = N(n-1)/(1+cff);
    P(n) = P(n-1) + cff*N(n);
end
```

10.22