ECES T580: Lab 4

Bhautik (Brian) Amin

Contents

Lab 3.1.1

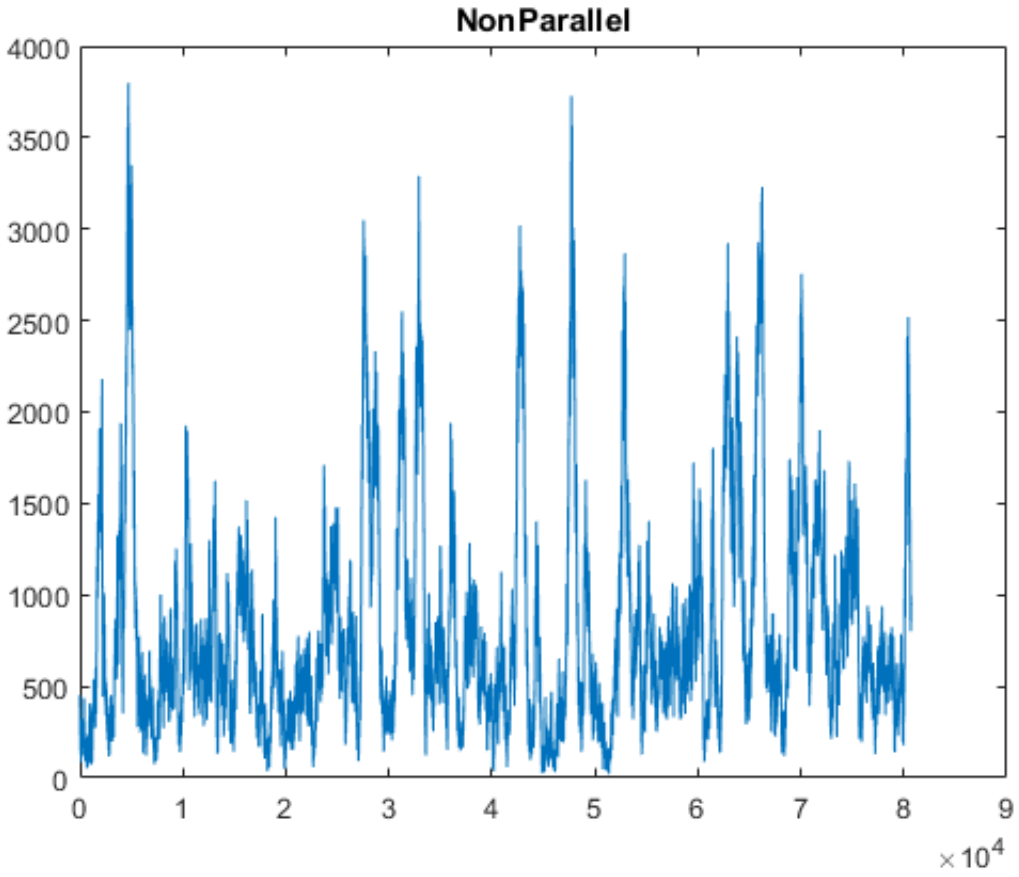
1. How could we break out the function to perform smaller computations? We could break up the data into segments, and perform our necessary operations amongst these segments in parallel. This will reduce the amount of computation time. To break up threebasefreq_stft into smaller computations, we can segment the DNA into chunks, and spin the FFT loop into its own thread/worker. These chunks can be processed and put back together when all of the workers have completed the task

```
hbb = genbankread('hbb_region_chr11.gb'); % Load data
```

Without Parallel Programming Running previous function and getting time stamp for benchmark

```
disp('Non-Parallel method:')
tic
Threebaseperiodicity_vs_position_old=threebasefreq_stft(hbb.Sequence,1000,1024);
toc
figure(1)
plot(Threebaseperiodicity_vs_position_old)
title('NonParallel')
```

```
Non-Parallel method:
Elapsed time is 13.657401 seconds.
```



With Parallel Programming

2. Modify threebasefreq_stft.m into a new function that allows for the parallelized computation

The function I created is called parallel_threebasefreq_stft (Located in Appendix). Function takes the same inputs, along with two new inputs: division, and index. See appendix for comments on the development of the function

3. Use the new code to parallelize the previous job

```
% delete(gcp('nocreate')) % Clear previous workers and set up new pool

stft_position = {}; % Initialize cell array to score processed chunks (Will be out of order)
worker_num = 2; % Setup for 2 core computer
%parpool(worker_num); % Setup worker pool
disp('Parallelized method')
tic % Begin timer
parfor i=1:worker_num
stft_position{i} = parallel_threebasefreq_stft(hbb.Sequence,1000,1024,worker_num,i);
end
```

```
Parallelized method
```

Piece the results from 4 workers together.

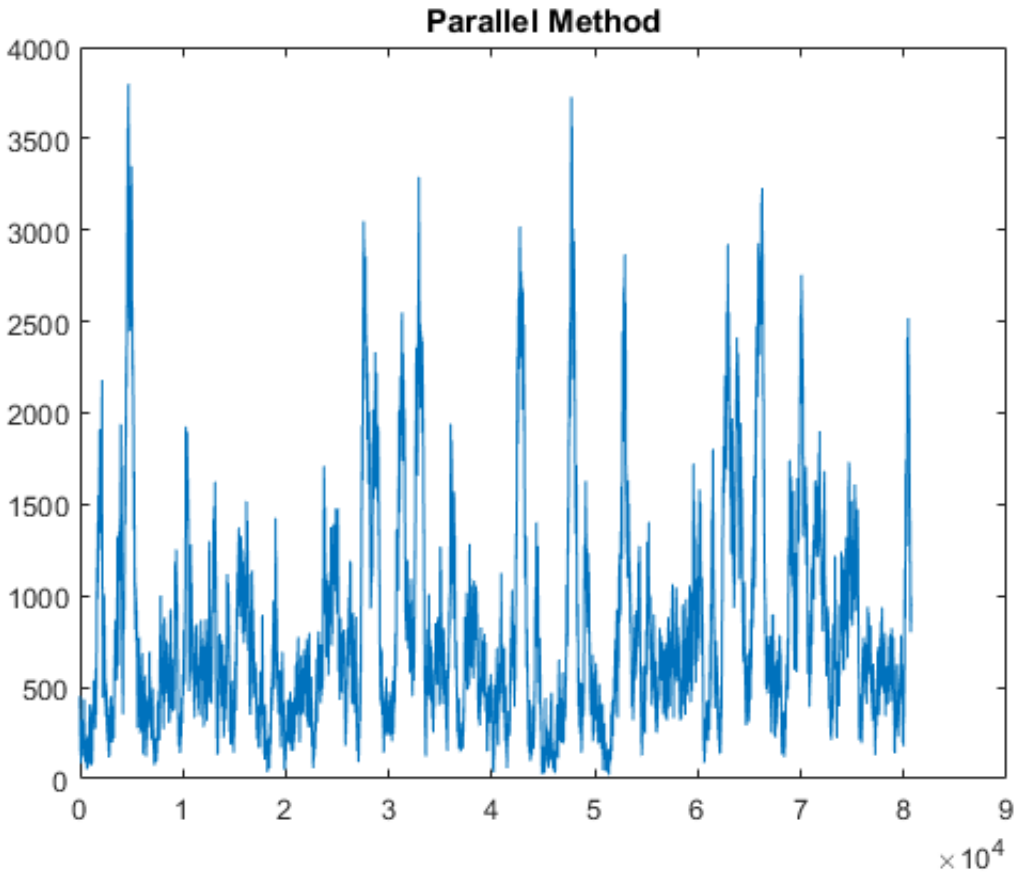
```
ordered_data = []; % Use struct for ordered data
for i=1:worker_num
    ordered_data = [ordered_data, stft_position(i)];
end
toc
```

```
Elapsed time is 9.917904 seconds.
```

Lab 3.2.1

Plot results for comparision

```
ordered_data = cell2mat(ordered_data); % Convert to struct
figure(2)
plot(ordered_data)
title('Parallel Method')
```



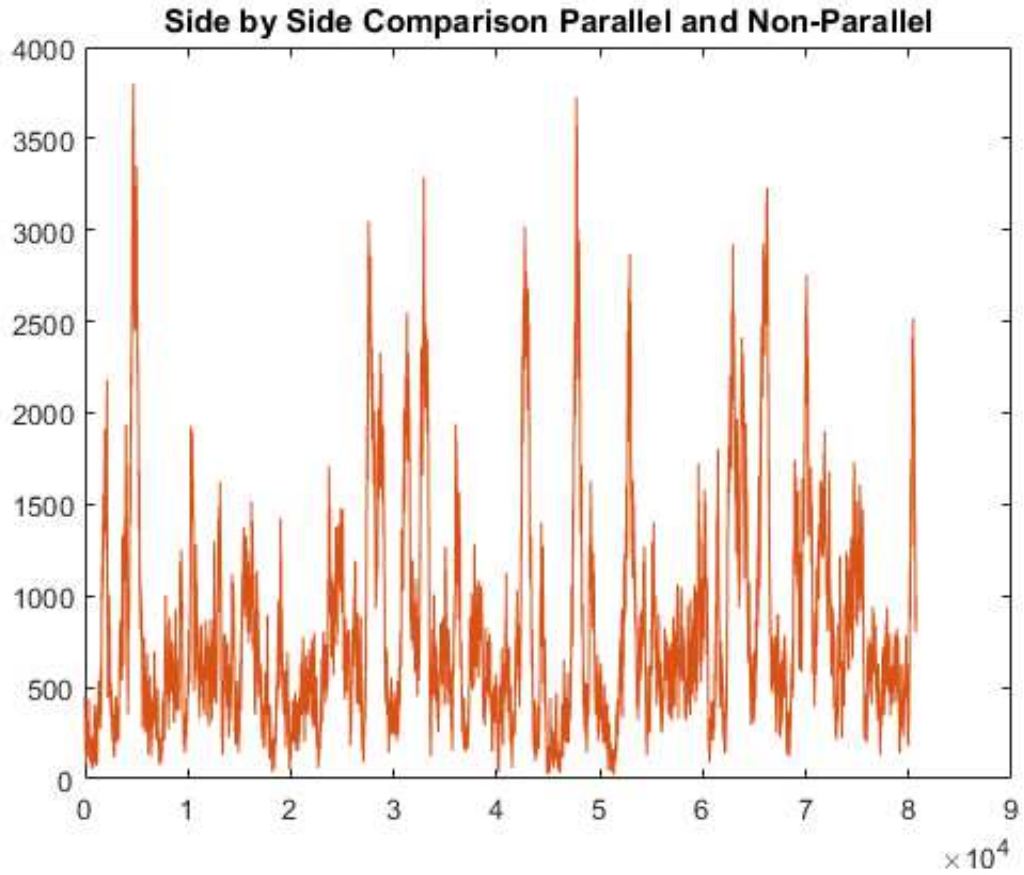
check difference:

```
diff = sum(abs(ordered_data - Threebaseperiodicity_vs_position_old));
disp('Absolute Sum Error:')
disp(diff)
```

```
Absolute Sum Error:
     0
```

Plot the results on top of each other

```
figure(3)
plot(Threebaseperiodicity_vs_position_old)
hold on;
plot(ordered_data)
title('Side by Side Comparison Parallel and Non-Parallel')
```



Function Appendix

```
function Threebaseperiodicity_vs_position = parallel_threebasefreq_stft (DNA_SEQUENCE, WINDOW
_LENGTH, NFFT, division, index)
% Function Goals:
% Divide DNA into a sequence of parts. For each parallelized iteration, the
% function needs to know which part is which and figure out which
% subsequence it will need to extract (start and end indices). Make sure
% all possible window positions are covered by the subsequence


% Calculate the amount of chunks desired (Depends on number of available
% workers)
chunks_size = floor(length(DNA_SEQUENCE)/division); %Floor used to round down number
```

```
% Determine start point based on index
start_point = (index-1)*chunks_size+1;


% If the index is the last segment of DNA, otherwise seg the end point with
% respect to the index and desired chunk size
if index == division
    end_point = length(DNA_SEQUENCE);
else
    end_point = index*chunks_size+WINDOW_LENGTH-1;
end

% Extract desired DNA sequence now
seq_div = DNA_SEQUENCE(start_point:end_point);
% Run the individual segment through threebasefreq_stft and return results
Threebaseperiodicity_vs_position = threebasefreq_stft (seq_div, WINDOW_LENGTH, NFFT);
end
```

Published with MATLAB® R2017a