Section 2 Problem

In this problem, you will optimize two codes. The goal is to reduce the running time of each code. You will get points based on the percentage of the running time you have reduced.

Here are things to look for when optimizing:

(1) numeric arrays have less memory overhead than structure or cell arrays

(2) use clear to delete variables no longer needed

(3) vectorize loops whenever possible

(4) logical indexing much faster than using find to extract index arrays

(5) delayed copy can make a huge difference (when large matrix passed to function, don?t unnecessarily modify it)

(6) sparse matrices can be much more efficient than full matrices, in terms of both storage and operations.

(7) Preallocation of numeric arrays, rather then incrementally growing their size within a loop.

Some, but not necessarily all of these optimizations will be significant in this problem. MATLAB?s profiler is a useful tool in determining bottlenecks.

Task 1

Optimize the code in `optimize_me.m` and in `optimize_me_too.m`. Profile the code using MATLAB?s profiler before and after optimization.

`Feel free to use the code box below.`

```
%Write your code here%
```

Task 2

Run the code before the optimization for 10 times and find the average running time. Run the code after the optimization for 10 times and find the average running time. Find the ratio of the average running time after the optmization and before the optimization. This ratio should be less than 1.

Feel free to use the box code below.

```
%Write your code here%
```

Checkpoint

Please answer the following questions and put the answers in the EdX page:

(A) When profiling |optimize_me.m,|on what line of the code is the majority of time spent?

(B) On line 14 of `optimize_me.m`, the grid is being defined. Is there a way to vectorize this and write this in one line of code?

(C) On line 25 of |optimize_me.m,|a difference matrix is created. Is it possible to use sparse storage instead?

(D) Line 33 of `optimize_me.m |shows a simple recursion relation between the new value of |U |and its old value. What is the memory efficient storage of |U`?

(E) On what line of |optimize_me_too.m |is a delayed copy being made?

(F) Think about the linear algebra operation being done on line 33 of `optimize_me.m |and in |optimize_me_too.m. |Give the code expression to eliminate the need of |optimize_me_too.m. |This effectively avoids making a delayed copy and explicitly forming the identity matrix. Enter your expression in the format |U = x - x*x*x*x.`

(G) What optimizations were the most significant for `optimize_me.m`?

(H) What is the ratio you get from Task 2? Round it to the nearest thousandth.