MATLAB

wrap-up

CS 101 lec28

2018-05-02
Roadmap

The Last Day
Are you present today?

A  Yes.
B  Yes.
C  Yes.
D  Yes.
E  Yes.
Image Processing
Partial Differential Equations
Boundary-value problems: elliptic PDEs

\[ \nabla^2 u(x, y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \]
Types of PDEs

- **Initial-value problems: parabolic PDEs**

\[
\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}
\]
Initial-value problems: hyperbolic PDEs

\[ \frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \]
Types of PDEs

\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)
\]

\[
\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}
\]

\[
\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}
\]
Your machine has many processors, so it can solve certain kinds of problems *in parallel*.

The simplest way is to use `parfor`—parallel for.

```matlab
parfor i = 1:12
disp( i );
end
```
Notice that the processes don’t always execute in the same order—this is known as race conditions.

To keep this from being a problem, parallel algorithms must be constructed carefully.
Performance

- Application: a computationally intensive task
- Serial version (one process):

```matlab
tic
n = 200;
A = 500;
a = zeros(n);
for i = 1:n
    a(i) = max(abs(eig(rand(A))));
end
toc
```
Application: a computationally intensive task

Parallel version (many processes):

```matlab
tic
n = 200;
A = 500;
a = zeros( n );
parfor i = 1:n
    a( i ) = max( abs( eig( rand( A ) ) ) );
end
toc```
The first time you run a parallel process in MATLAB, it is slower because the system must be started up in parallel. Later runs are faster.

Not all problems are suitable for parallel execution because of this setup time. This one is too small:

```matlab
tic
n = 2 .^ 10;
A = zeros(n);
for i = 1:n  % try parfor also
    A(:,i) = 1:n .* sin(i*2*pi/2.^10);
end
toc
```
Practice Question
Compose a MATLAB script which prints out each number from 1 to 100. For each number, the following set of rules should be applied to the output:

- If divisible by 3, print number followed by "fizz".
- If divisible by 5, print number followed by "buzz".
- If divisible by both 3 and 5, print number followed by both "fizz" and "buzz".
- Otherwise, just print the number.
For instance, the expected output from 9 to 15 will be:
9fizz
10buzz
11
12fizz
13
14
15fizzbuzz
for i = (1:100)
    if mod(i,15) == 0
        str = sprintf( '%dfizzbuzz',i );
    elseif mod(i,3) == 0
        str = sprintf( '%dfizz',i );
    elseif mod(i,5) == 0
        str = sprintf( '%dbuzz',i );
    else
        str = sprintf( '%d',i );
    end
    disp( str );
end
What Else is There?
Other things to know

Python concepts:
- data types: set, objects (classes)
- SciPy (scientific functions, equation solvers)
- Pandas (data analysis)
- SymPy (symbolic algebra)
- functional programming: map, filter, lambda
Other things to know

MATLAB concepts:
- toolboxes: Image Processing, Statistics, PDEs
- differential equation solvers (ode45)
- parallel programming (parfor)
- cells (like Python lists)
- linear algebra, symbolic algebra
Are you present today?

A  Yes.
B  Yes.
C  Yes.
D  Yes.
E  Yes.