MATLAB with Modelling and Parallel Computing

May 11, 2025

Ocean University of China, High Performance Computing Club, Qingdao 266100

Mathematical Modelling Introduction

- Mathematical modelling is the process of representing real-world problems using mathematical concepts and equations.
- It involves identifying the key variables, relationships, and constraints that govern the system being studied.
- The goal is to create a simplified representation of the problem that can be analyzed and solved using mathematical techniques.

Some Examples of Mathematical Modelling

- Differential Equations: Used to model dynamic systems, such as population growth, chemical reactions, and fluid flow.
- Optimization Problems: Used to find the best solution among a set of feasible solutions, such as minimizing costs or maximizing profits.
- Statistical Models: Used to analyze and interpret data, such as regression analysis, time series analysis, and hypothesis testing.
- Simulation Models: Used to simulate complicated systems, such as weather forecasting, traffic flow, and financial markets.

All these examples could be well solved by MATLAB, we will give some simple examples of these.

Differential Equations

To begin with, simple differential equations could be solved by hand, but complicated ones might not have analytical solutions. In this case, we could use numerical methods to solve them.

MATLAB has a built-in function `ode45` which is a powerful tool for solving ordinary differential equations (ODEs) numerically.

Also consider Partial Differntial Euqations, finite element methods could be used and it is simple to achieve in MATLAB.

But still remember that MATLAB could also solve the symbolic equations, which is also useful in theoritical analysis.

Optimization Problems

Optimization problems are common in many fields, the fundamental idea is to find the best solution, this is quite familiar with find the minimum or maximum of a function.

MATLAB provides a variety of optimization functions, such as `linprog` and `intlinprog`, which can be used to solve linear optimization problems.

Statistical Models

Statistical models are very hot in data science, while they are used to analyze and interpret data, doing the analysis is typically the baisis of machine learning and large language models.

These models are large than thought and basically all needed for applications in almost all fields. And MATLAB could also doing very well to fix it.

Some further ideas could be related to probability distributions, regression analysis, and hypothesis testing.

Simulation Models

Simulation models are much more related to applications, which not matched our topic today. But still, we could use MATLAB to simulate complicated systems, such as weather forecasting, traffic flow, and financial markets.

A fantastic tool is called Simulink, which can be used to model and simulate dynamic systems.

Parallel Computing

Parallel computing is a powerful technique that allows us to solve large and complicated problems more efficiently by dividing the workload among multiple processors or cores.

MATLAB provides several built-in functions and toolboxes for parallel computing, such as the Parallel Computing Toolbox, which allows us to take advantage of multi-core processors and clusters.

Some examples of parallel computing in MATLAB include parallel for loops, distributed arrays, and GPU computing.

Parallel Computing Example

Here we only explain some basic ideas of parallel computing, and we will give a simple example to compare the efficiency when using parallel computing and not in MATLAB.

```
f = @(x) exp(-x.^2); % Define the function
a = 0; b = 1; % Define the limits
n = 1000000; % Number of intervals
h = (b - a) / n; % Calculate the width
sum1 = 0; % Initialize the sum
for i = 1:n
    x = a + (i - 0.5) * h;
    sum1 = sum1 + f(x);
end
result1 = h * sum1; % Calculate the result
```

Parallel Computing Example

This is the another example using parallel computing, which is similar to the previous one. The only difference is that we use `parfor` instead of `for`.

```
parpool; % Start a parallel pool
f = @(x) exp(-x.^2); % Define the function
a = 0; b = 1; % Define the limits
n = 1000000; % Number of intervals
h = (b - a) / n; % Calculate the width
sum2 = 0; % Initialize the sum
parfor i = 1:n % Use parfor for parallel loop
    x = a + (i - 0.5) * h;
    sum2 = sum2 + f(x);
end
result2 = h * sum2; % Calculate the result
```

Parallel Computing Example

Now we want to compare the efficiency of these two examples, we could use `tic` and `toc` to measure the time taken by each example.

```
tic; % Start the timer
% First codes
toc; % Stop the timer and display the time taken

tic; % Start the timer
% Second codes
toc; % Stop the timer and display the time taken
```

Parallel Computing Example

By comparing the result, you can find that the second example is much faster than the first one, especially when the number of intervals is large.

In future works, we would introduce parallel computing further, thus you could use it in your own work.

Actually, for MATLAB or Python, parallel computing is much more important than we thought, and it is not only used in numerical methods, but also in machine learning and deep learning.

For most cases, these interpreted languages have serious performance issues, so parallel computing is much more important to help fix this problem.