|  |  |  |  |
| --- | --- | --- | --- |
| Weekly Research Report | | | |
| Name | Aiyung | Duration | 2024/12/19 ~ 2024/12/25 |
| Date | 2024/12/25 | (week 15) | |

* Bring your research notebook every time for cross check when present your weekly report.
* The weekly report should be written over 1 page.

1. Brief title of this report (本報告主題)

C++數值求解函式庫的選用實驗

1. Research issue address at … (研究過程中發現的問題)

有眾多Library，需要時間分析其運算能力選擇合適的來使用。

1. Method or possible solutions (提出可能的解決方法)

簡單設計數值實驗觀察各自運算效率如何。

* 1. 大型有限差分方程
  2. 大型稀疏矩陣求反運算

1. Outcomes and new derivative problems (因應該方法產生的結果，及或衍生的新問題)

表格 1 不同求解器求解不同維度有限差分方程時間(s)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 矩陣長寬 (N-by-B grid) | | |
|  |  | N=100 | N=150 | N=200 |
| 求解器 | Fastor | 0.10465 | 0.64483 | 1.98235 |
| Blaze | 0.11872 | 0.62767 | 2.15301 |
| Eigen | 0.12378 | 0.72485 | 2.17740 |
| Armadillo | 0.39592 | 2.08010 | 5.90168 |
| XTensor | 0.66842 | 3.24171 | 10.06010 |
| MATLAB | 1.96156 | 7.86771 | 20.87833 |

一張含有 文字, 螢幕擷取畫面, 繪圖, 圖表 的圖片

自動產生的描述

圖 1不同求解器求解不同維度有限差分方程時間圖

在N=200時，迭代次數為55376。

表格 2 利用不同求解器求取的稀疏矩陣耗時

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 求解器 | Fastor | Blaze | Eigen | Armadillo | XTensor | MATLAB |
| 時間(s) | 14.38638 | 18.44755 | 15.45984 | 62.20703 | 77.25874 | 135.24481 |

Armadillo、Fastor、Eigen都有和MATLAB類似的操作，其中Armadillo最接近（也最短）。

但Armadillo在求解時還要呼叫LAPACK所以編譯時間和求解時間都長一點。

Faster專注於處理「固定大小」的陣列，在此情況下會很快。

XTensor是設計給機器學習用的，雖然有針對多維數組進行優化但對大型反矩陣不那麼擅長。由於是設計給Python使用者，被譽為C++的NumPy因此聲量才很大。

Blaze同上，但支援函數很少。

1. Conclusion & Discussions (小結與討論)

雖然不是最快，考慮到安裝方便性、易讀性與移植性，最後決定使用Eigen。

（只需要引入該標頭檔即可，不用安裝任何東西。）

少有的MPL授權，可以在使用Eigen的情況下閉源個人開發程式。

1. Plan for next week (下周預期工作內容，提出可能解決本周問題的幾種規劃)

可能會先集中在期末報告的製作。

附錄：實驗用MATLAB code

tic;

run\_finite\_difference(200);

toc;

function run\_finite\_difference(num)

  err = 2;

  iter = 0;

  x = (0:num-1)\*pi/(num-1);

  u = zeros(num, num);

  u(:, 1) = sin(x);

  u(:, num) = sin(x)\*exp(-pi);

  while (iter < 100000 && err > 1e-6)

    [u,err] = finite\_difference\_seq\_impl(u);

    iter = iter + 1;

  end

  disp(['Error : ', num2str(err)]);

  disp(['Iterations: ', num2str(iter)]);

end

function [u,err] = finite\_difference\_seq\_impl(u)

  num = size(u, 1);

  u\_old = u;

  u(2:num-1, 2:num-1) = ...

    ((u\_old(1:num-2, 2:num-1) + u\_old(3:num, 2:num-1) + ...

    u\_old(2:num-1, 1:num-2) + u\_old(2:num-1, 3:num) )\*4.0 + ...

    u\_old(1:num-2, 1:num-2) + u\_old(1:num-2, 3:num) + ...

    u\_old(3:num, 1:num-2) + u\_old(3:num, 3:num) ) / 20.0;

  err = sqrt(sum(sum((u - u\_old).^2)));

end