

qda_linear_solver 模块文档

i 我们基于 QPanda (C++) 实现了 3 个线性求解器函数

```
// 理想的绝热演化求解器
VectorXcd linear_solver_ideal(MatrixXcd A, VectorXcd b);
// 严格符合赛题诸多要求限制的求解器
VectorXcd linear_solver_contest(MatrixXcd A, VectorXcd b);
// 为精度做了各种优化的求解器
VectorXcd linear_solver_ours(MatrixXcd A, VectorXcd b);
```

ideal

linear_solver_ideal 实现了基于理想哈密顿模拟的绝热演化求解器，其中时间演化算子 e^{-iHt} 直接由数学计算得到，通过 matrix_decompose() 来获取对应的量子逻辑门线路实现。基本程序结构如下：

```
1 // 绝热演化参数
2 const int S = 200; // 总演化阶段步数
3 const int T = 10; // 单步哈密顿量模拟时间
4 // 制备系统初态 |b>
5 qcir << amplitude_encode(qv, amplitude);
6 // 制备含时哈密顿量 H_s
7 H_s = (1 - s) * H0 + s * H1
8 // 绝热演化
9 for (int s = 1; s <= S; s++) {
10 // 含时哈密顿量近似为不含时
11 MatrixXcd H = H_s(float(s) / S);
12 // 时间演化算子的矩阵形式
13 MatrixXcd iHt = dcomplex(0, -1) * H * T;
14 MatrixXcd U_iHt = iHt.exp();
15 // 矩阵形式转化为量子逻辑门线路
16 qcir << matrix_decompose(U_iHt, qv);
17 }
18 // 概率测量读取振幅，解出 |x>
19 QProg qprog = createEmptyQProg() << qcir;
20 qvm.directlyRun(qprog);
```

contest

`linear_solver_contest` 实现了基于近似哈密顿模拟的绝热演化求解器，其中时间演化算子 e^{-iHt} 通过一阶近似为 $e^{-iHt} \approx I - iHt$ ，通过 BlockEncoding 技术获得其酉矩阵版本，再通过 `matrix_decompose()` 来获取对应的量子逻辑门线路实现。基本程序结构如下，与 `linear_solver_ideal` 的主要差别是 12 ~ 16 行的近似。

```
1 // 绝热演化参数
2 const int S = 200; // 总演化阶段步数
3 const int T = 1; // 单步哈密顿量模拟时间
4 // 制备系统初态 |b>
5 qcir << amplitude_encode(qv, amplitude);
6 // 制备含时哈密顿量 H_s
7 H_s = (1 - s) * H0 + s * H1
8 // 绝热演化
9 for (int s = 1; s <= S; s++) {
10 // 含时哈密顿量近似为不含时
11 MatrixXcd H = H_s(float(s) / S);
12 // 时间演化算子的一阶近似
13 MatrixXcd iHt = exp_iHt_approx(H, T);
14 // 谱范数规范化 & 进行块编码
15 iHt = normalize_QSVT(iHt);
16 MatrixXcd U_iHt = block_encoding_QSVT(iHt);
17 // 近似的时间演化算子转化为量子逻辑门线路
18 qcir << matrix_decompose(U_iHt, qv);
19 }
20 // 概率测量读取振幅，解出 |x>
21 QProg qprog = createEmptyQProg() << qcir;
22 qvm.directlyRun(qprog);
```

ours

`linear_solver_ours` 基于 `linear_solver_contest` 迭代改造而成，引入了大量 trick 来提升结果的保真度，探索绝热演化线性求解器的能力上限，主要包含下列 trick：

- 更多的演化阶段步数 S ，更长的物理演化时间 T (第 2 ~ 3 行)
- AQC(P) 调度函数 (第 7 行)
- e^{-iHt} 二阶近似 (第 13 行)
- 特征滤波 EF (第 20 ~ 24 行)

```

1 // 绝热演化参数
2 const int S = 300; // 总演化阶段步数
3 const int T = 10; // 单步哈密顿量模拟时间
4 // 制备系统初态 |b>
5 qcir << amplitude_encode(qv, amplitude);
6 // 制备含时哈密顿量 H_s, 使用 AQC(P=2) 调度函数 f(s)
7 H_s = (1 - f(s)) * H0 + f(s) * H1
8 // 绝热演化
9 for (int s = 1; s <= S; s++) {
10 // 含时哈密顿量近似为不含时
11 MatrixXcd H = H_s(float(s) / S);
12 // 时间演化算子的二阶近似
13 MatrixXcd iHt = exp_iHt_approx(H, T, 2);
14 // 谱范数规范化 & 进行块编码
15 iHt = normalize_QSVT(iHt);
16 MatrixXcd U_iHt = block_encoding_QSVT(iHt);
17 // 近似的时间演化算子转化为量子逻辑门线路
18 qcir << matrix_decompose(U_iHt, qv);
19 }
20 // 制作特征滤波矩阵 & 进行块编码
21 MatrixXcd EF = EF_R_l(H1);
22 MatrixXcd U_EF = block_encoding_QSVT(EF);
23 // 特征滤波矩阵转化为量子逻辑门线路
24 qcir << matrix_decompose(U_EF, qv);
25 // 概率测量读取振幅, 解出 |x>
26 QProg qprog = createEmptyQProg() << qcir;
27 qvm.directlyRun(qprog);

```

附录

基准单元测试运行结果参考 (T=1000):

 LS_unittest

case-study experiment

对比实验

在 [playground/qda_linear_solver.py](#) 和 [playground/vis_eigen_filter_ls.py](#) 中我们也实现了 Randomization Method, vanillan AQC, AQC(P), AQC(EXP) 和 Eigen Filter 作为对比参考。

解示例方程组:

$$\begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

| implementation | solution | fidelity | comment |
|----------------|----------------------------------|-----------------|----------------|
| LS_ideal | [-0.722392, -0.691484] | 0.999761 | S=200, T=S*10 |
| LS_contest | [-0.777313, 0.629114] | 0.104793 | S=200, T=S*1 |
| LS_contest | [-0.882665, -0.470002] | 0.95648 | S=200, T=S*2 |
| LS_contest | [-0.692988, -0.720949] | 0.999805 | S=200, T=S*4 |
| LS_contest | [-0.683247, -0.730187] | 0.999449 | S=200, T=S*10 |
| LS_contest | [-0.6704, -0.742] | 0.998718 | S=300, T=S*4 |
| LS_ours | [-0.630297, -0.776354] | 0.994653 | S=200, T=S*2 |
| LS_ours | [-0.709877, -0.704326] | 0.999992 | S=200, T=S*4 |
| LS_ours | [-0.706603, -0.70761] | 0.999999 | S=400, T=S*2 |
| vanilla AQC | [-0.6143, -0.7846] | 0.989147 | S=200, T=19799 |
| AQC(p=1.001) | [-0.6839, -0.7295] | 0.999437 | S=200, T=1027 |
| AQC(p=1.5) | [-0.7174, -0.6965] | 0.999745 | S=200, T=1027 |
| AQC(p=2) | [-0.7002, -0.7139] | 0.999960 | S=200, T=1027 |
| AQC(exp) | [-0.655, -0.7467] | 0.991093 | S=200, T=12959 |
| QDA | [0.621482, 0.783429] | 0.993422 | S=500 |
| RM (algo 1) | [0.5049, 0.4860, 0.4967, 0.5120] | 0.997524 | S=310 |
| RM (algo 2) | [0.5090, 0.5133, 0.4881, 0.4885] | 0.999385 | S=310 |

❗ Note that target solution is [0.7071, 0.7071] for all AQC-based methods (up to a global phase), while [0.5, 0.5, 0.5, 0.5] for RM-based ones

对比实验: AQC 神话

| implementation | solution | fidelity | comment |
|----------------|--------------------|----------|---------------|
| AQC(p=1.001) | [-0.6850, -0.7284] | 0.999436 | S=310, T=1027 |

| implementation | solution | fidelity | comment |
|----------------|--------------------|----------|---------------|
| AQC(p=1.25) | [-0.7221, -0.6917] | 0.999707 | S=310, T=1027 |
| AQC(p=1.5) | [-0.7162, -0.6976] | 0.999745 | S=310, T=1027 |
| AQC(p=1.75) | [-0.7028, -0.7112] | 0.999878 | S=310, T=1027 |
| AQC(p=2) | [-0.7007, -0.7135] | 0.999959 | S=310, T=1027 |
| AQC(v-func) | [-0.7030, -0.7111] | 0.999923 | S=310, T=1027 |

消融实验

TODO: C++ 反复编译实在太烦人，再说吧.....

i QSVT 与 ideal 精度持平，即使使用了 approx(-iHt) 近似；ARCSIN 比 QSVT 差在逐渐积累的浮点误差

☐ $f(s) = \text{linear}$

| fidelity | solution | run config |
|----------|------------------------|------------------------------------|
| 0.116225 | [-0.784498, 0.620131] | ideal; S=200, T=1 |
| 0.999883 | [-0.696206, -0.717842] | ideal; S=200, T=10 |
| 0.999964 | [-0.701095, -0.713068] | ideal; S=300, T=10 |
| 0.116241 | [-0.784508, 0.620118] | QSVT; S=200, T=1 |
| 0.999527 | [-0.685027 -0.728517] | QSVT; S=200, T=10 |
| 1.000000 | [-0.707475, -0.706739] | QSVT; S=300, T=10 |
| 0.999998 | [-0.708495, -0.705716] | QSVT; S=2000, T=1 |
| 0.640244 | [0.9959, -0.090459] | ARCSIN; S=200, T=1 |
| 0.873111 | [0.962113, 0.272652] | ARCSIN; S=200, T=1, norm(lmbd*iHt) |
| 0.321988 | [0.897129, -0.441769] | ARCSIN; S=2000, T=1 |

☐ $f(s) = \text{poly}(2)$

| fidelity | solution | run config |
|-----------------|------------------------|-------------------------------------|
| 0.856842 | [-0.970448, -0.241309] | QSVT; S=200, T=1 |
| 0.999872 | [-0.718318, -0.695715] | QSVT; S=200, T=10 |
| 0.999988 | [-0.71061, -0.703587] | QSVT; S=300, T=10 |
| 1.000000 | [-0.70689, -0.707324] | QSVT; S=2000, T=1 |
| 0.828373 | [0.981853, 0.189643] | ARCSIN; S=200, T=1 |
| 0.489276 | [0.962659 -0.270718] | ARCSIN; S=200, T=1, norm(iHt) |
| 0.983201 | [0.824295, 0.56616] | ARCSIN; S=200, T=1, norm(lmbd*iHt) |
| 0.998871 | [0.672715, 0.739901] | ARCSIN; S=2000, T=1 |
| 0.917811 | [0.929724, 0.368257] | ARCSIN; S=2000, T=1, norm(lmbd*iHt) |

by Armit
2024/5/30