1 From PDE to matrix $Ax = b$

1.1 Chosen equation

We start with the PDE that we want to solve. For this example, we start with the heat equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1}$$

Where T is the temperature, t is the time, α is the thermal diffusivity and x is the position. The value of α depends on the properties of the material whose temperature you want to measure:

$$\alpha = \frac{\kappa}{c\rho} \tag{2}$$

Since these properties do not depend on time or position, we can leave out the thermal conductivity (κ), heat capacity (c) and density (ρ). For now, at least.

1.2 Preconditions

Let's establish the preconditions of our little example. Our example contains a copper wire of length L and initial temperature $T_{initial}$, which is heated, by temperature T_h, and cooled, by temperature T_c, on opposite sides of the wire. For simplicity, we'll assume that our wire is infinitely thin and infinitely short. This means that our wire has only one dimension, making it much easier to calculate. For visualisation purposes, our example looks like this:

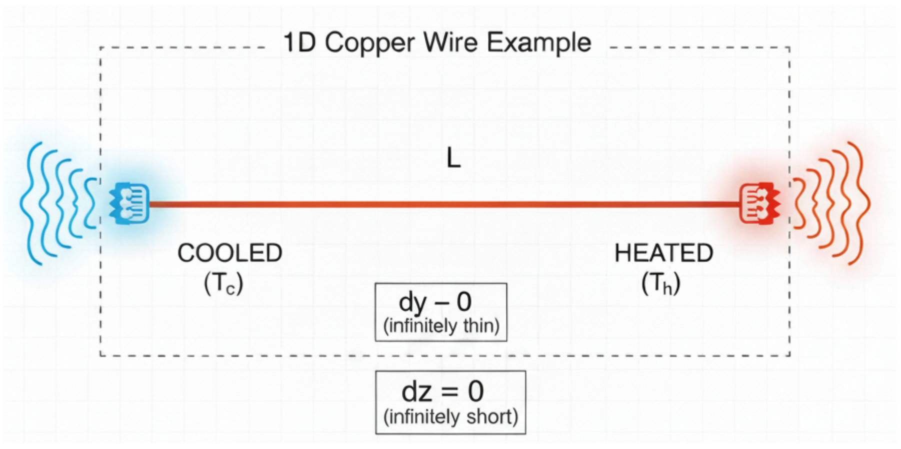

Figure 1: Visualization of example

We'll call the position where the wire is cooled $x = 0$ and the position where the wire is heated $x = L$. This creates a difference in length to calculate over. When we measure 8 points along the length of the wire, waiting 10 seconds after applying the temperatures before taking the measurements, the following array and variables are obtained:

1.3 Distretization of x and t

Suppose we take N measurements, each of which is determined by an index $i = 0, 1, ..., N-1$. When we discretize the space, we obtain the following equation:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{(T_i - T_{i+1})^2}{(dx)^2} = \frac{T_i - 2T_{i+1} + T_{i+2}}{(dx)^2} \tag{3}$$

Where dx is the space step size, which is determined by $dx = L/(N-1)$.

Now, we will also discretize the time using the backward Euler method. The Backward Euler equation is as follows:

$$\frac{\partial T_i}{\partial t} \approx \frac{T_i^{(n+1)} - T_i^{(n)}}{dt} \tag{4}$$

Here, n is the time index, determined by $n = 0, 1, ..., N-1$, and dt is the time step size, determined by $dt = t/t_{step}$. Here, t is the amount of time waited before measuring the whole wire, and t_{step} is the time step for the number of points measured. Combining the last two equations yields the following:

$$\frac{T_i^{(n+1)} - T_i^{(n)}}{dt} = \alpha \frac{T_i^{(n+1)} - 2T_{i+1}^{(n+1)} + T_{i+2}^{(n+1)}}{(dx)^2} \tag{5}$$

1.4 Matrix formulation

Let's introduce the constant s. This constant is determined by the following equation:

$$s = \frac{\alpha dt}{(dx)^2} \tag{6}$$

Introducing s allows us to simplify our equation:

$$T_i^{(n+1)} - T_i^{(n)} = s(T_i^{(n+1)} - 2T_{i+1}^{(n+1)} + T_{i+2}^{(n+1)}) \tag{7}$$

This equation can be rewritten as follows:

$$T_i^n = (1+2s)T_{i+1}^{(n+1)} - sT_i^{(n+1)} - sT_{i+2}^{(n+1)} \tag{8}$$

This involves isolating all the T^n terms from the $T^{(n+1)}$ terms. This gives matrix A via the equation $A \cdot T^{(n+1)} = T^n$. When this equation is written in matrix form, the following matrix is formed:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -s & 1+2s & -s & 0 & \cdots & 0 & 0 \\ 0 & -s & 1+2s & -s & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -s & 1+2s & -s \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

Matrix A will be an $N \times N$ matrix.

Once matrix A has been determined, it is time to determine vector b. While A contains information about the time and properties of the wire, b contains details of the initial temperature conditions. It is a simple 1D array expressed as follows: $b = T_c, T_{initial}, T_{initial}, ..., T_{initial}, T_h$. where b has length N.

Now, A and b are in a linear form can be computed using a linear solver. For example, the numpy function $np.linalg.solve()$ from Numpy can be used. This is also how a linear PDE equation is classically solved using matrices.

2 Classical Result

When we compute our example using the method described above, with the following variables: $L = 1$ m, $N = 8$, $t = 10$ s, $t_{step} = 0.1$ s, $T_h = 323$ K, $T_c = 223$ K, $T_{initial} = 273$ K, the result will look like this:

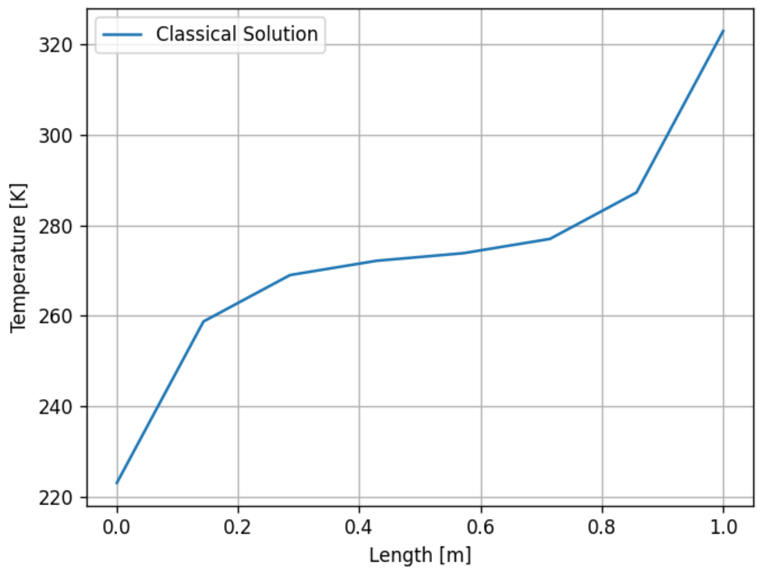

Figure 2: Classical result of example

Of course, this example disregards the thermal effects occurring outside the wire. Therefore, this example is not very realistic.

3 Quantum Solution

3.1 Finding quantum A and b

To find the quantum solution, we will create a Variational Quantum Linear Solver (VQLS), which will amplitude-encode our temperatures in a quantum state. The probability amplitudes of these states will therefore represent the temperatures at the end. A common trick during amplitude encoding is to apply Hadamard gates to all qubits in order to represent all possible solutions. During this process, the number of possible states will increase from 1 state to 2^n states. Here, n is the number of qubits in the circuit. In our example we have an 8×8 matrix, so we will need $log_2(8) = 3$ qubits.

Now, let's solve the problem with the quantum gates. First, we need to quantify our linear heat equation as follows: $A\left|x\right\rangle \approx \left|b\right\rangle$ so that it can be solved using a quantum circuit. As A is a linear system, it can be divided into several components. Our use case will determine the combination:

$$A = c_0A_0 + c_1A_1 + c_2A_2 + c_3A_3 \tag{10}$$

Here, A_n is part of the full matrix, with c_n as its coefficients. A_n must be represented using quantum gates, but the coefficients of c_n must be determined manually. Matrix A can be recreated with quantum gates using the operators of the quantum gates. For matrix A, we find the following combination:

$$A = -sX_0 - sX_1 - sX_2 + (1 + 2s)I \tag{11}$$

Here X_n is the Pauli-X quantum gate acting on the n^{th} qubit. I is an identity gate which leaves the qubit's state unchanged. The equation shows that the coefficients $c_n = [-s, -s, -s, 1 + 2s]$ are used. The following matrix can be found when matrix A is computed using linear operators and their coefficients:

$$A = \begin{pmatrix} 1+2s & -s & 0 & -s & 0 & -s & 0 & 0 \\ -s & 1+2s & -s & 0 & -s & 0 & 0 & 0 \\ 0 & -s & 1+2s & -s & 0 & 0 & 0 & -s \\ -s & 0 & -s & 1+2s & 0 & 0 & -s & 0 \\ 0 & -s & 0 & 0 & 1+2s & -s & 0 & -s \\ -s & 0 & 0 & 0 & -s & 1+2s & -s & 0 \\ 0 & 0 & 0 & -s & 0 & -s & 1+2s & -s \\ 0 & 0 & -s & 0 & -s & 0 & -s & 1+2s \end{pmatrix} \tag{12}$$

This is not the same as the classical matrix, but for the purposes of this tutorial, we need a linear combination that can be computed fairly quickly. Feel free to create a more suitable matrix using quantum gates!

Once the linear combination has been found, the next step is to create the state $\left|b\right\rangle$. As this state represents the quantum version of the classical solution's vector b. To create the state $|b\rangle$, we need a normalized version of the vector b. Once this has been determined, the values can be assigned to the amplitudes of the quantum states[1]. If we take vector b from the classical solution, state $|b\rangle$ will be:

$$|b\rangle = U_x |0\rangle = 0.288\,|000\rangle + 0.352\,|001\rangle + 0.352\,|010\rangle + 0.352\,|011\rangle + 0.352\,|100\rangle + 0.352\,|101\rangle + 0.352\,|110\rangle + 0.417\,|111\rangle \tag{13}$$

Here, U_x is the operation that is applied to the state $|0\rangle$ to get the state $|b\rangle$.

3.2 Cost function

Once A and b have been determined, we can calculate x as follows:

$$|b\rangle \approx \frac{A\,|x\rangle}{\sqrt{\langle x|\,A^{\dagger}A\,|x\rangle}} \tag{14}$$

Here, A is the matrix created using the quantum gates from earlier, and $|b\rangle$ is the state created from earlier. The solution to our problem is given by the state $|x\rangle$. This state must be read out using quantum measurements, which are inherently probabilistic. Rather than relying on costly overlap measurements each time, we introduce a trainable layer of R_y rotations (a variational ansatz). A classical optimizer tunes the rotation angles so that the prepared state $|x(\theta)\rangle$ approximates the desired solution. After training, we can prepare state $|x(\theta^*)\rangle$ directly and obtain the solution via standard sampling, thus avoiding the need for repeated Hadamard-overlap estimates. Adding these rotations will change state $|x\rangle$ to:

$$|x\rangle = V(\theta)\,|0\rangle \tag{15}$$

In order to find the optimal rotation, the overlap between the state $|\psi\rangle = A\,|x\rangle$ and the state $|b\rangle$ must be determined. This can be done using the following formula:

$$C_G = 1 - |\,\langle b|\psi\rangle\,|^2 = 1 - \frac{|\,\langle b|\psi\rangle\,|^2}{||\,|\psi\rangle\,||^2} \tag{16}$$

Where C_G is the global cost or overlap between the two states $|b\rangle$ and $|\psi\rangle$ and $||\,|\psi\rangle\,||$ is the norm of the state $|\psi\rangle$. Once the maximum overlap has been reached, the value of state $|\,\langle b|\psi\rangle\,|$ will equal one, indicating that the optimal rotation has been found.

In order to determine the maximum overlap between the two states $|b\rangle$ and $|\psi\rangle$, we need to perform a Hadamard test. How the Hadamard test works will be explained shortly, but first it is important to find the test's equation. The only issue with the global cost function is that it depends on the full inner product of two multi-qubit states, which is a global quantity. Such global overlaps are extremely difficult to measure directly using current NISQ hardware, because the device can only access the local expectation values of simple observables (e.g. Pauli operators) rather than the global properties of the full

[1]The normalization was necessary to fulfil the state rule: $\sum_{k=0}^{n-1} \alpha_k^2 = 1$.

state.

For this reason, the global cost function must be rewritten into a sum of local, measurable quantities. This is done by expressing the overlap in terms of Pauli-Z measurements using the commonly used overlap operator:

$$P = \frac{1}{2} + \frac{1}{2n} \sum_{j=0}^{n-1} Z_j \tag{17}$$

Here, P represents the overlap and Z_j is a Pauli-Z operator applied to the j^{th} qubit. Substituting this equation into equation **??** yields the following result:

$$C = \frac{1}{2} - \frac{1}{2n} \frac{|\sum_{j=0}^{n-1} \langle b|Z_j|\psi\rangle|^2}{|| \, |\psi\rangle \, ||^2} \tag{18}$$

We can rewrite this as:

$$\langle b|\psi\rangle = \langle b| A \cdot V(\theta) |0\rangle = \langle 0| U_x^\dagger A \cdot V(\theta) |0\rangle = \sum_l c_l \langle 0| U_x^\dagger A_l V(\theta) |0\rangle \tag{19}$$

Since A is a matrix, it can be rewritten as $A = \sum_l c_l A_l$. This explains the derivation used in equation **??**. Squaring the upper equation gives:

$$|\langle b|\psi\rangle|^2 = \sum_{l,\nu} c_l c_\nu^* \langle 0| V(\theta)^\dagger A_\nu^\dagger U_x U_x^\dagger A_l V(\theta) |0\rangle \tag{20}$$

Substituting this equation in equation **??** yields the following equation:

$$C = \frac{1}{2} - \frac{1}{2n} \frac{\sum_{l,\nu,j} c_l c_\nu^* \langle 0| V(\theta)^\dagger A_\nu^\dagger U_x Z_j U_x^\dagger A_l V(\theta) |0\rangle}{\sum_{l,\nu} c_l c_\nu^* \langle 0| V(\theta)^\dagger A_\nu^\dagger A_l V(\theta) |0\rangle} \tag{21}$$

The denominator is not derived here, but it can be derived using the following formula: $|\psi\rangle^2 = (A \cdot V(\theta) |0\rangle)^2$. In this way, the originally unmeasurable global cost function is converted into a locally measurable cost function suitable for the current NISQ hardware through the Hadamard test.

The final cost function **??** provides us with the Hadamard test function, as we can measure the following coefficients:

$$H_{test} = \langle 0| A_\nu^\dagger U_x Z_j U_x^\dagger A_l V(\theta) |0\rangle \tag{22}$$

Where $A_\nu^\dagger$ is the conjugate transpose of matrix A_l, which is part of the linear combination in equation **??**, but with the coefficients set to ν. Equation **??** does not require the adjoint variational operator, which can interfere with machine learning.

3.3 Hadamard test

In quantum computing, a Hadamard test is used to store information during a specific process. Here, we will use it to determine the degree of overlap between the states. A Hadamard test uses an ancilla qubit, which is a qubit that is not used in the main quantum circuit, but which is used to store information. This qubit enters a Hadamard gate, which creates a 50/50 superpositioned state. Then, the ancilla qubit is given the ability to store information about the process by applying the controls from certain control quantum gates to the ancilla qubit. When another Hadamard gate is applied to the ancilla qubit at the end, the number of measurements of the state $|0\rangle$ tells you the degree of overlap.

Looking back at equation **??**, we can compute all the operators as explained earlier. However, this does not apply to the A_l and $A_{l'}$ operators. Since we want to apply our linear combinations to the ancilla qubit in order to preserve the overlap, we need to create a controlled version of the linear combination we determined earlier. The control will be applied to the ancilla qubit and the operation to the desired qubit.

If we train our circuit using a variety of different rotations and then measure the ancilla qubit, we can find the optimal solution to our PDE. Of course, this assumes that enough epochs are used[2].

4 Quantum Result

When you apply the quantum solution in code. First, we must denormalise the results since all quantum states are normalised. When we denormalise the amplitudes using the classical vector b and plot the resulting vector in a figure, we get the following result:

[2]The amount of times the code went trough the whole training dataset.

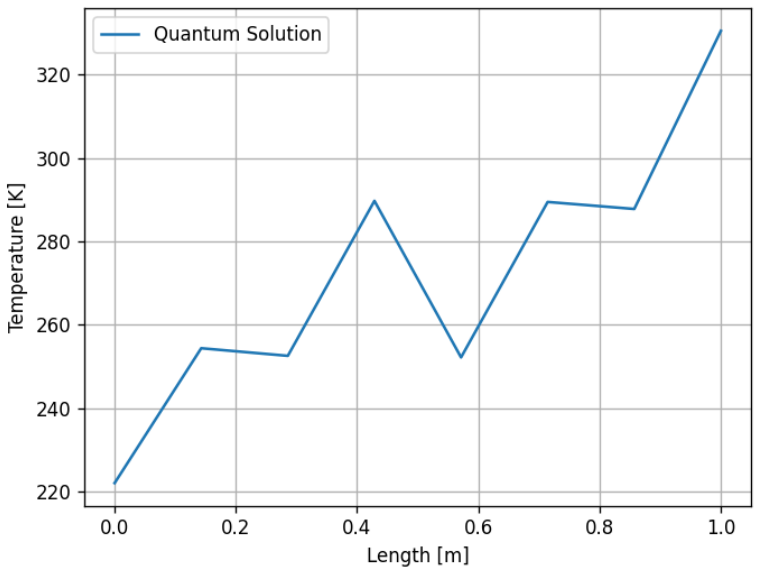

Figure 3: Quantum result of example

This still excludes thermal effects from outside the wire. Which still doesn't make the example that realistic.

5 Classical vs Quantum

When the classical and quantum results are plotted together, the following can be seen:

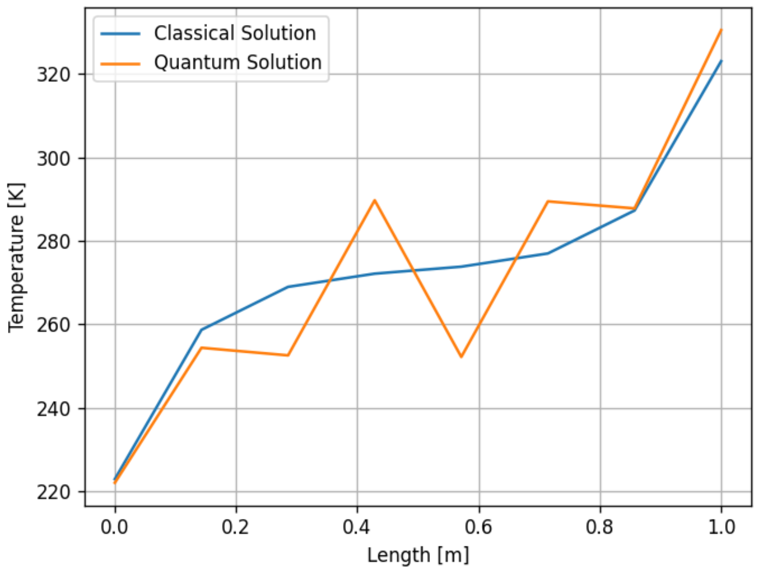

Figure 4: Quantum vs classical result of example

This shows that the results are quite similar, but not completely overlapping.