Maastricht University

UNIVERSITY OF KWAZULU-NATAL
CENTRE FOR QUANTUM TECHNOLOGY

Mark Fingerhuth

Quantum-enhanced machine learning: Implementing a quantum k-nearest neighbour algorithm

Bachelor Thesis

In partial fulfillment of the requirements for the degree Bachelor of Science (BSc) at Maastricht University.

Supervision

Prof. Francesco Petruccione
Dr. Fabrice Birembaut

January 2017

Preface

Blah blah ...

Contents

Abstract

Blah blah ...

Nomenclature

Symbols

$\otimes$	Tensor product	
i	Imaginary unit	$i = \sqrt{-1}$
$\dagger$	Hermitian conjugate	Complex conjugate transpose

Indicies

a	Ambient
air	Air

Acronyms and Abbreviations

QML	Quantum machine learning
QC	Quantum computer
Prob	Probability
CNOT	Controlled NOT gate
CCNOT	Controlled controlled NOT gate
CU	Controlled U gate (where U can be any unitary quantum gate)

Chapter 1

Introduction

SOME MORE GENERAL INTRO TEXT HERE?

1.1 Research Question

Matthias Troyer citation about q software engineering needed?

Chapter 2

Theoretical Foundations and Methods

2.1 Quantum Bits

2.1.1 Single Qubit Systems

Classical computers manipulate bits, whereas quantum computer's most fundamental unit is called a quantum bit, often abbreviated as qubit. Bits as well as qubits are binary entities, meaning they can only take the values 0 and 1. A classical non-probabilistic bit can only be in one of the two possible states at once. In contrast, qubits obey the laws of quantum mechanics, which gives rise to the powerful property that - besides being a definite 0 or 1 - they can also be in a superposition of the two states. Mathematically this is expressed as a linear combination of the states 0 and 1:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

where α and β are complex coefficients ($\alpha, \beta \in \mathbb{C}$) and are often referred to as phase factors or amplitudes. $|0\rangle$ is the Dirac notation for the qubit being in state 0 and it represents a two-dimensional vector in a complex 2-D vector space (called Hilbert space $\mathcal{H}_2$). $|0\rangle$ and $|1\rangle$ are the computational basis states and they constitute an orthonormal basis of $\mathcal{H}_2$. For the sake of clarity, $|0\rangle$ and $|1\rangle$ can be thought of as the 2-D vectors shown below.

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{2.2}$$

Subbing these vectors into Equ. 2.1 yields the vector representation of $|\psi\rangle$:

$$|\psi\rangle = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix} \tag{2.3}$$

However, even though a qubit can be in a superposition of $|0\rangle$ and $|1\rangle$, when measured it will take the value $|0\rangle$ with a probability of

$$Prob(|0\rangle) = |\alpha|^2 \tag{2.4}$$

and $|1\rangle$ with a probability of

$$Prob(|1\rangle) = |\beta|^2 \tag{2.5}$$

The fact that the probability of measuring a particular state is equal the absolute value squared of the respective amplitude is called Born's rule (citation). Since the total probability of measuring any value has to be 1, the following normalization condition must be satisfied:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.6}$$

Therefore, a qubit is inherently probabilistic but when measured it collapses into a single classical bit (0 or 1). It follows that a measurement destroys information about the superposition of the qubit (the values of α and β). This constitutes one of the main difficulties when designing quantum algorithms since only limited information can be obtained about the final states of the qubits in the quantum computer.

Similar to logic gates in a classical computer, a QC manipulates qubit by means of quantum logic gates which will be introduced in detail in Section 2.2. Generally, an arbitrary quantum logic gate U acting on a single qubit state is a linear transformation given by a 2x2 matrix whose action on $|\psi\rangle$ is defined as:

$$U|\psi\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a*\alpha + b*\beta \\ c*\alpha + d*\beta \end{pmatrix} \tag{2.7}$$

Matrix U must be unitary which means that a) its determinant is equal to unity and b) its Hermitian conjugate $U^\dagger$ must be equal to its inverse. These properties are expressed in Equ. 2.8 and 2.9 below. All quantum logic gates must be unitary since this preserves the normalization of the qubit state it is acting on.

$$a) \mid det(U) \mid = 1 \tag{2.8}$$

$$b) \quad UU^\dagger = U^\dagger U = \mathbb{1} = UU^{-1} = U^{-1}U \tag{2.9}$$

Using spherical polar coordinates, a single qubit can be visualized on the so-called Bloch sphere by parameterising α and β in Equ. 2.1 as follows:

$$|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.10}$$

The Bloch sphere has a radius of 1 and is therefore a unit sphere. The $|0\rangle$ qubit state is defined to lie along the positive z-axis ($\hat{z}$) and the $|1\rangle$ state is defined to lie along the negative z-axis ($-\hat{z}$) as labelled in Fig. 2.1. At this point, it is important to note that these two states are mutually orthogonal in $\mathcal{H}_2$ even though they are not orthogonal on the Bloch sphere.

Qubit states on the Bloch equator such as the $\hat{x}$ and $\hat{y}$ coordinate axes represent equal superpositions where $|0\rangle$ and $|1\rangle$ both have measurement probabilities equal to 0.5. The $\hat{x}$-axis for example represents the equal superposition $|q\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. As illustrated in Fig. 2.1 any arbitrary 2-D qubit state $|\psi\rangle$ can be decomposed into the polar angles θ and ϕ and visualized as a vector on the Bloch sphere. Such an object is called the Bloch vector of the qubit state $|\psi\rangle$. The Bloch sphere will be the main visualization tool for qubit manipulations in this thesis.

[1]Reprinted from Wikipedia, n.d., Retrieved September 7, 2016, from `https://en.wikipedia.org/wiki/Bloch_Sphere`. Copyright 2012 by Glosser.ca. Reprinted with permission.

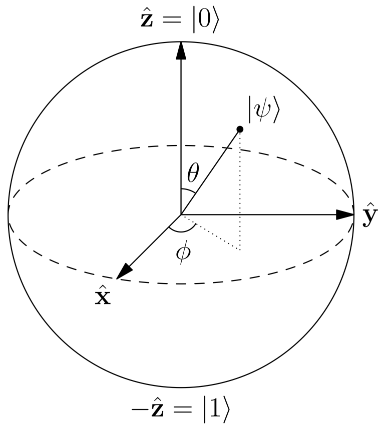


Figure 2.1: An arbitrary two-dimensional qubit $|\psi\rangle$ visualized on the Bloch sphere.[1]

2.1.2 Multiple Qubit Systems

When moving from single to multi qubit systems a new mathematical tool, the so-called tensor product (symbol $\otimes$), is needed. A tensor product of two qubits is written as:

$$|\psi\rangle \otimes |\psi\rangle = |0\rangle \otimes |0\rangle = |00\rangle \tag{2.11}$$

whereby the last expression omits the $\otimes$ symbol which is the shorthand form of a tensor product between two qubits.

In vector notation a tensor product of two vectors (red and green) is defined as shown below:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1 * \begin{pmatrix}1\\0\end{pmatrix}\\ 0 * \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \tag{2.12}$$

The last expression in Equ. 2.12 shows that the two-qubit state $|00\rangle$ is no longer two but four-dimensional. Hence, it lives in a four-dimensional Hilbert space $\mathcal{H}_4$. A quantum gate acting on multiple qubits can therefore not have the same dimensions as a single-qubit gate (Equ. 2.7) which demands for a new gate formalism for multi qubit systems.

Consider wanting to apply an arbitrary single-qubit gate U (Equ. 2.7) to the first qubit and leaving the second qubit unchanged, essentially applying the identity matrix $\mathbb{1}$ to it. To do this, one defines the tensor product of two matrices as follows,

$$U \otimes \mathbb{1} = \begin{pmatrix}a & b\\c & d\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \begin{pmatrix}a * \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & b * \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ c * \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & d * \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}a & 0 & b & 0\\0 & a & 0 & b\\c & 0 & d & 0\\0 & c & 0 & d\end{pmatrix} \tag{2.13}$$

Thus, the result of the tensor product $U \otimes \mathbb{1}$ is a unitary 4x4 matrix which can now be used to linearly transform the 4x1 vector representing the $|00\rangle$ state in Equ. 2.12:

$$U \otimes \mathbb{1} |00\rangle = \begin{pmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ c & 0 & d & 0 \\ 0 & c & 0 & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ c \\ 0 \end{pmatrix} \tag{2.14}$$

One can also first perform the single qubit operations on the respective qubits followed by the tensor product of the two resulting vectors:

$$(U \otimes \mathbb{1})(|0\rangle \otimes |0\rangle) = U |0\rangle \otimes \mathbb{1} |0\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ c \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ c \\ 0 \end{pmatrix} \tag{2.15}$$

This formalism can be extended to any number of qubits and

2.1.3 Entanglement

Introduce entanglement and non-factorising tensor states here!

2.2 Quantum Logic Gates

In order to perform quantum computations, tools, analogous to the classical logic gates, are needed for qubit manipulation. Quantum logic gates are square matrices that can be visualized as rotations on the Bloch sphere. The following subsections will introduce the major single and multi qubit logic gates.

2.2.1 Single Qubit Gates

2.2.2 Multiple Qubit Gates

Controlled NOT Gate

The most important two-qubit quantum gate is the controlled NOT or CNOT gate given by the following 4x4 matrix:

$$CNOT = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.16}$$

The CNOT gate takes two qubits, control and target qubit, as input. If and only if the control qubit is in the $|1\rangle$ state, the NOT (X) gate is applied to the target qubit. In equations, the CNOT will always be followed by parantheses containing the control qubit followed by the target qubit (e.g. CNOT(0,1)). The input-output relation for the CNOT gate is given in Table 2.2 below.

To demonstrate the usefulness of the CNOT gate consider starting with two unentangled qubits both in the $|0\rangle$ state,

$$|\phi\rangle = |0\rangle \otimes |0\rangle = |00\rangle \tag{2.17}$$

Applying the H gate onto the first qubit yields the following (still unentangled) state:

$$(H \otimes \mathbb{1})\,|\phi\rangle = (H \otimes \mathbb{1})\,|00\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle \tag{2.18}$$

Now consider applying the CNOT to this state whereby the control qubit is coloured red and the target qubit green.

$$CNOT(0,1) \otimes (\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}(\mathbb{1} \otimes X)\,|10\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle \tag{2.19}$$

The last expression in Equ. 2.19 is one of the famous Bell states which are four maximally entangled states. Thus, this example demonstrates how the CNOT gate is crucial for the generation of entangled states since it applies the X gate to a qubit depending on the state of a second qubit.

Toffoli Gate

The most important three-qubit gate is a controlled controlled NOT (CCNOT) quantum gate which is often referred to as the Toffoli gate. It is defined by the following 8x8 matrix:

$$Toffoli = CCNOT = \begin{pmatrix} \mathbb{1}_6 & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.20}$$

where $\mathbb{1}_6$ is the 6x6 identity matrix.

The Toffoli gate takes three inputs specified in parantheses, first the two control qubits and lastly the target qubit. If and only if both control qubits are in the $|1\rangle$ state the X gate is applied to the target qubit. Toffoli gates are usually decomposed into quantum circuits with CNOT and single-qubit gates in order to keep the set of quantum gates small (Shende & Markov, 2008).

nCNOT Gate

The nCNOT gate is the generalization of the CNOT and the Toffoli gate. It takes n control qubits and one target qubit as input and if and only if all control qubits are in the $|1\rangle$ state the X gate is applied to the target. The nCNOT matrix representation is given by:

$$nCNOT = \begin{pmatrix} \mathbb{1}_{2^n-2} & 0 \\ 0 & X \end{pmatrix} \tag{2.21}$$

In practise, nCNOT gates are usually not implemented directly but decomposed into larger quantum circuits consisting only of CNOT and single-qubit gates (Nielsen & Chuang, 2010). However, such decompositions will not be relevant for this work.

2.3 Classical Machine Learning

2.3.1 k-nearest Neighbour Algorithm

Use text from proposal and slightly adapt.

2.4 Methods (own chapter?)

2.4.1 Programming environment

2.4.2 IBMs QC

Table 2.1: Table of major single-qubit quantum logic gates.

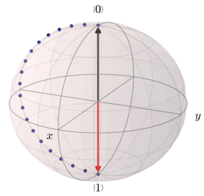
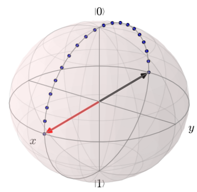
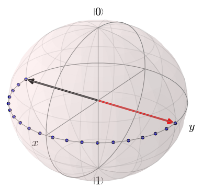
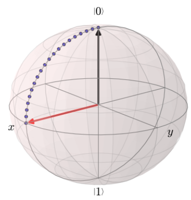
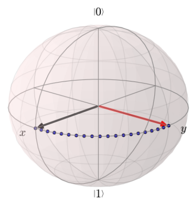
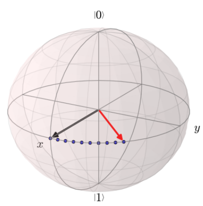
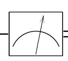
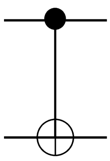
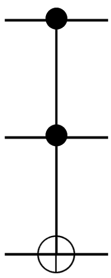
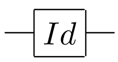
Gate	Name	Circuit representation	Matrix	Description	Rotation	Bloch sphere
$\mathbb{1}$	Identity	$\boxed{Id}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Idle or waiting gate	-	-
X	Qubit flip	$\boxed{X}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Swaps amplitudes of $\|0\rangle$ and $\|1\rangle$	π rotation around $\hat{x}$	
Y	Qubit & phase flip	$\boxed{Y}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Swaps amplitudes and introduces phase	π rotation around $\hat{y}$	
Z	Phase flip	$\boxed{Z}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Adds a negative sign to the $\|1\rangle$ state	π rotation around $\hat{z}$	
H	Hadamard	$\boxed{H}$	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$	Creates equal superpositions	$\frac{\pi}{2}$ rotation around $\hat{y}$ and π rotation around $\hat{x}$	
S	$\frac{\pi}{2}$ rotation gate	$\boxed{S}$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\sqrt{Z}$	$\frac{\pi}{2}$ rotation around $\hat{z}$	
S†	$-\frac{\pi}{2}$ rotation gate	$\boxed{S^\dagger}$	$\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$	Adjoint of S	$-\frac{\pi}{2}$ rotation around $\hat{z}$	
T	$\frac{\pi}{4}$ rotation gate	$\boxed{T}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$	$\sqrt{S}$	$\frac{\pi}{4}$ rotation around $\hat{z}$	
T†	$-\frac{\pi}{4}$ rotation gate	$\boxed{T^\dagger}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{-\frac{i\pi}{4}} \end{pmatrix}$	Adjoint of T	$-\frac{\pi}{4}$ rotation around $\hat{z}$	
ZM	Z-basis measurement		-	Measurement in standard basis	Collapses the state	-

Table 2.2: CNOT truth table with first qubit as control, second qubit as target.

Input	Output
00	00
01	01
10	11
11	10

Table 2.3: Table of major multi-qubit quantum logic gates.

Gate	Name	Circuit representation	Matrix	Description
CNOT	Controlled NOT		$\begin{pmatrix} \mathbb{1} & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	CNOT(c_1, target)
Toffoli	Controlled controlled NOT		$\begin{pmatrix} \mathbb{1}_6 & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	CCNOT(c_1, c_2, target)
nCNOT	n-controlled NOT		$\begin{pmatrix} \mathbb{1}_{2^n-2} & 0 \\ 0 & X \end{pmatrix}$	nCNOT(c_1,..,c_n, target)

Chapter 3

Quantum-enhanced Machine Learning

3.0.1 Quantum k-nearest Neighbour Algorithm

3.0.2 Quantum State Preparation Algorithms

Grover & Rudolph

Diffusion matrix from quantum random walks

Trugenberger et al.

Schack paper Maria mentioned but which I haven't used and looked at.

Chapter 4

Working with LaTeX

This chapter explains how to typeset some of the most common elements contained in a technical report using LaTeX.

4.1 Headings

Your report can be structured using several different types of headings. Use the commands `\chapter{.}`, `\section{.}`, `\subsection{.}`, and `\subsubsection{.}`. Use the asterisk symbol `*` to suppress numbering of a certain heading if necessary, for example, `\section*{.}`.

4.2 References and Footnotes

References to literature are included using the command `\cite{.}`. For example (?, ?, ?). Your references must be entered in the file `bibliography.bib`. Making changes or adding new references in the bibliography file can be done manually or by using specialized software such as *JabRef* which is free of charge.

Cross-referencing within the text is easily done using `\label{.}` and `\ref{.}`. For example, this paragraph is part of chapter 4; more specifically section 4.2 on page 13. You will need to compile your document twice in order for the cross-referencing to be updated.

Footnotes[1] are added using the command `\footnote{.}`, but try to avoid the used of footnotes altogether.

4.3 Lists

Three types of list-environments are commonly used: `itemize`, `enumerate`, and `description`. The following example uses `itemize` to create a list without numbering

- point one; and
- point two

created using

[1] The use of footnotes is generally not recommended.

```
\begin{itemize}
  \item point one; and
  \item point two
\end{itemize}
```

The following example uses `enumerate` to create a list with numbering

1. point one; and
2. point two

created using

```
\begin{enumerate}
  \item point one; and
  \item point two
\end{enumerate}
```

The following example uses `description` to create a list with custom text as bullet-points

P1 point one; and

P2 point two

created using

```
\begin{description}
  \item[P1] point one; and
  \item[P2] point two
\end{description}
```

4.4 Tables

Table 4.1 shows an example of a simple table-layout. Try to avoid vertical lines on tables. The Internet contains countless resources on how to create special elements and structures in tables such as cells spanning multiple rows, rotated text, sideways tables, justification of cell elements, etc.

Table 4.1: Driving cycle data of ECE-15, EUDC, and NEDC.

Description	Unit	ECE	EUDC	NEDC
Duration	s	780	400	1180
Distance	km	4.052	6.955	11.007
Average velocity	km/h	18.7	62.6	33.6
Idle speed	%	36	10	27

This table was created using

```
\begin{table}[ht]
\begin{center}
\caption{Driving cycle data of ECE-15, EUDC, and NEDC.}\vspace{1ex}
\label{tab:table}
\begin{tabular}{llccc}\hline
Description & Unit & ECE & EUDC & NEDC \\ \hline
Duration & s & 780 & 400 & 1180 \\
Distance & km & 4.052 & 6.955 & 11.007 \\
Average velocity & km/h & 18.7 &  62.6 & 33.6 \\
```

```
Idle speed & \% & 36 & 10 & 27 \\ \hline
\end{tabular}
\end{center}
\end{table}
```

Table 4.2 shows a more advanced version of Tab. 4.1 using the `booktabs` package. Inspect the source code of this document to see how this was done.

Table 4.2: Driving cycle data of ECE-15, EUDC, and NEDC.

		Driving cycle		
Description	Unit	ECE	EUDC	NEDC
Duration	s	780	400	1180
Distance	km	4.052	6.955	11.007
Average velocity	$^{km}/_{h}$	18.7	62.6	33.6
Idle speed	%	36	10	27

4.5 Working with Units

The package `\usepackage{units}` enables two useful commands, namely `\unit[.]{.}` and `\unitfrac[.]{.}{.}`. Use these commands to display units in a concise way, for example

$$\delta t = 1\,\mathrm{s} \tag{4.1}$$
$$v = 5\,{}^{\mathrm{m}}/_{\mathrm{s}}. \tag{4.2}$$

This example was done using

```
\begin{align}
\delta t &= \unit[1]{s}\\
v &= \unitfrac[5]{m}{s}.
\end{align}
```

4.6 Including Graphics

It is recommended that you only use encapsulated post-script graphics `.eps` in your report. If you mix `.eps` with other formats such as `.png`, `.jpeg` or `.gif`, you will most likely not be able to compile your report without errors. Note that figures created in MATLAB are easily saved in `.eps` format.

The inclusion of a figure can be done in the following way:

```
\begin{figure}[ht]
   \centering
   \includegraphics[width=0.75\textwidth]{img/k_surf.eps}
   \caption{Example of a figure.}
   \label{img:k_surf}
\end{figure}
```

Two figures are displayed next to each other using

```
\begin{figure}[ht]
  \begin{minipage}[t]{0.48\textwidth}
```

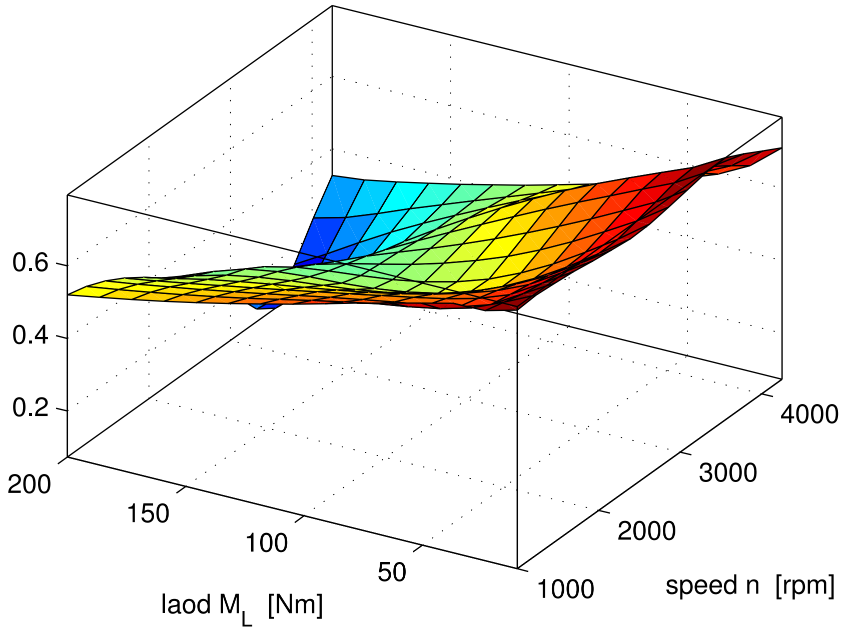


Figure 4.1: Example of a figure.

```
    \includegraphics[width = \textwidth]{img/cycle_we.eps}
  \end{minipage}
  \hfill
  \begin{minipage}[t]{0.48\textwidth}
    \includegraphics[width = \textwidth]{img/cycle_ml.eps}
  \end{minipage}
  \caption{Two figures next to each other.}
  \label{img:cycle}
\end{figure}
```

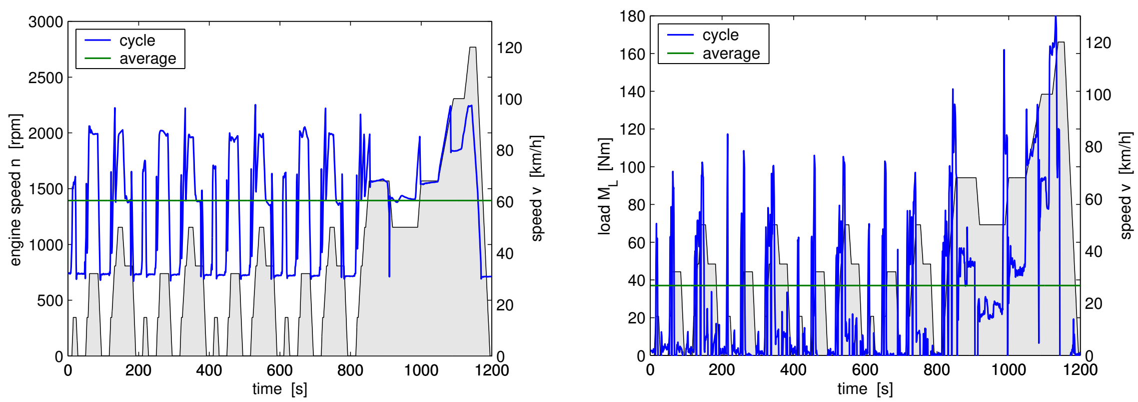


Figure 4.2: Two figures next to each other.

The positioning parameter `h` (here) forces your figure to be placed in the current position relative to your text. You may add `t` (top), `b` (bottom), and/or `p` (page) to allow for more flexible positioning within your document. For instance, `[tb]` forces your figure to be placed either on the top or bottom of a page.

4.7 Equations

The most common way to include equations is using the `equation` environment.

$$p_{\mathrm{me0f}}(T_e, \omega_e) \ = \ k_1(T_e) \cdot (k_2 + k_3 S^2 \omega_e^2) \cdot \Pi_{\mathrm{max}} \cdot \sqrt{\frac{k_4}{B}} \, . \qquad (4.3)$$

It is recommended to use `\mathrm{.}` for subscripts comprising more than two letters since it reduces the width of the subscript significantly and improves readability. The corresponding code is

```
\begin{equation}\label{eq:p_me0f}
 p_\mathrm{me0f}(T_e,\omega_e) \ = \ k_1(T_e) \cdot (k_2+k_3 S^2
 \omega_e^2) \cdot \Pi_\mathrm{max} \cdot \sqrt{\frac{k_4}{B}} \, .
\end{equation}
```

Equations, such as Eq. (4.3), may be referenced using `\eqref{.}`. In-line mathematical content is created using `$.$`, for example $a^2 + b^2 = c^2$. It is practically possible to typeset any equation in LaTeX. Equation (4.4) shows an example of a more advance structure.

$$x_n^k(i) = \left\{ \begin{array}{ll} y(i) & \text{if} \quad x_{n-1}^k(i) \leq \mathbf{x} \\ z(i) & \text{otherwise} \end{array} \right. , \text{for} \quad i = \{1, \ldots, N\}. \qquad (4.4)$$

4.8 Including Code in your Document

Include samples from your Matlab code using the `lstlistings` environment, for example

```
% Evaluate y = 2x
for i = 1:length(x)

  y(i) = 2*x(i);

end
```

This example was created using

```
\lstset{language=Matlab,numbers=none}
\begin{lstlisting}[frame=lines]
% Evaluate y = 2x
for i = 1:length(x)

  y(i) = 2*x(i);

end
\end{lstlisting}
```

where `\usepackage{mcode}` must be included in the preamble of your document. If you want to include the entire content of a file `mycode.m` in your document, simply input the path to `mycode.m` instead of pasting the entire content into your TeX-file

```
\lstset{language=Matlab,numbers=left}
\lstinputlisting{path/to/mycode.m}
```

Including the path to your m-file also ensures that the code in your report is always up-to-date. The `\lstset{language=Matlab}` command ensures that MATLAB syntax definitions are used, but many other languages are recognised as well such as `Fortran` and `C++`.

References

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information.* Cambridge university press.

Shende, V. V., & Markov, I. L. (2008). On the cnot-cost of toffoli gates. *arXiv preprint arXiv:0803.2316*.