JOURNAL OF THE MYANMAR ACADEMY OF ARTS AND SCIENCE

Mathematics and Computer Science

Vol. XVI, No.3, October 2018

Myanmar Academy of Arts and Science

Journal of the Myanmar Academy of Arts and Science

Vol. XVI, No.3

Contents

Section (i) Mathematics

Section (ii) Computer Studies

FINDING A NEW METHOD FOR THE SOLUTION OF NONLINEAR EQUATIONS $f(x) = 0$

Ohnmar Nwe*

Abstract

In this paper, the weak points of some numerical methods which can be used to find the solutions of nonlinear equations $f(x) = 0$ are introduced.Then the new method (OhnmarNwe's method) is presented. And also the convergence of the new method is proved and comparison of the convergence for the new method and Newton's Method are expressed. Finally, the weak point of the new method is discussed.

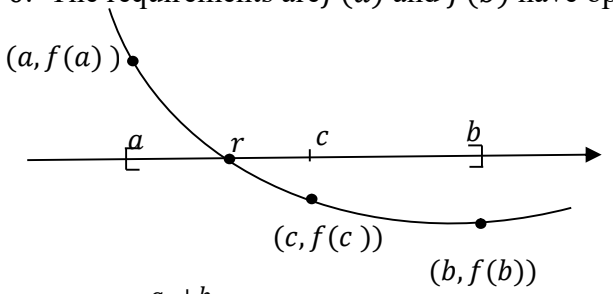
Introduction

We used the numerical methods for finding a zero of a continuous function. There are several methods, in which, we would like to discuss about the weak points of some numerical methods. We choose the methods are Bisection Method, Method of False Position, Secant Method, Newton – Raphson Method and Muller's Method. Our intention is to compare with the new method.

Weak Points of Some Methods

In this paper, we define the function f is continuous on the interval that we consider.

In **Bisection Method**, we need $f \in C[a, b]$ and to find $r \in [a, b]$ such that$f(r) = 0$. The requirements are$f(a)$ and $f(b)$ have opposite signs.



The formula is $c_n = \frac{a_n + b_n}{2}$for all n. Here only the average of the interval (i.e., $\frac{a_n + b_n}{2}$) is used.

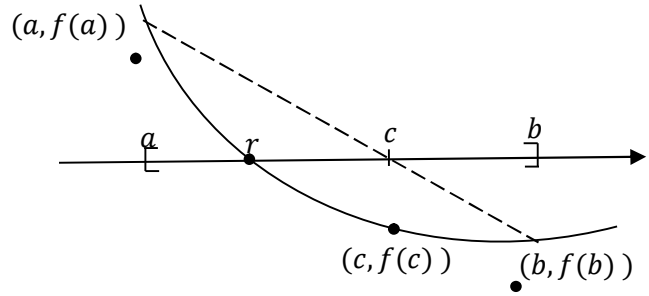
*. Dr., Associate Professor, Department of Mathematics, Loikaw University

The convergence of the method is base on the sense that

$$|r - c_n| \leq \frac{b-a}{2^{n+1}} \text{for} n = 0,1, \ldots \ldots$$

The weakness of this method is the convergence speed is fairly slow. If $f(x) = 0$ has several roots in $[a, b]$, it is not easy to calculate as a different starting points and intervals must be used to find each root.

In **Method of False Position**, $f(a)$ and $f(b)$ need to have opposite signs. It is used the line joining the points $(a, f(a))$and $(b, f(b))$.



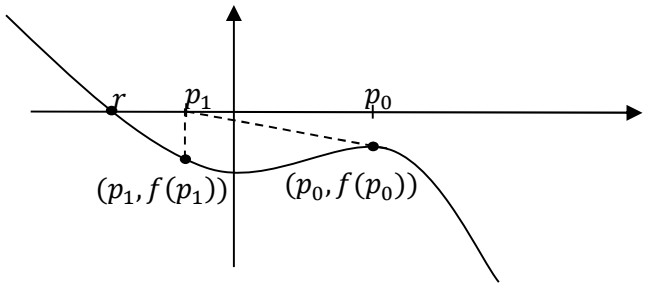
The formula is $c_n = b_n - \frac{f(b_n)(b_n - a_n)}{f(b_n) - f(a_n)}$for all n.The size of the$f(b_n)$ and the interval $(b_n - a_n)$ are used.

The convergence of this method is based on the sense that $(b_n - a_n) \to 0$ as $n \to \infty$.

This method is faster than the bisection method. But it is also not easy to calculate for the several roots in an interval.

The weak points of the Bisection Method and Method of False Position are they need two initial points which have opposite signs of function value. Also these methods are not so easy to find the several roots.

In **Newton-Raphson Method (Newton's Method)**, $f(x), f'(\mathrm{x})$ and $f''(\mathrm{x})$ need to be continuous near a root. This method used the tangent lines.

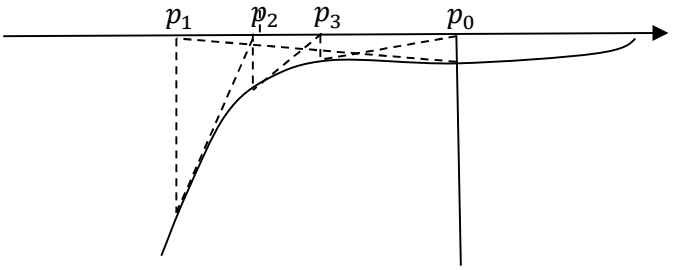


The formula is $p_k = \mathrm{p_{k-1}} - \frac{\mathrm{f(p_{k-1})}}{\mathrm{f'(p_{k-1})}}$for $k = 1,2, \ldots.$

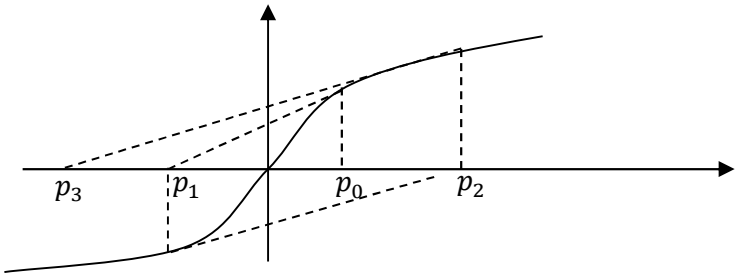
The convergence of this method is based on the Taylor polynomial and the fixed-point iteration.

This method is one of the most useful method. It is faster than the Bisection Method and Method of False Position. This method requires two function evaluations these are $f(p_{k-1})$and $f'(\mathrm{p_{k-1}})$. And have difficulty if $f'(\mathrm{p_{k-1}}) = 0$.

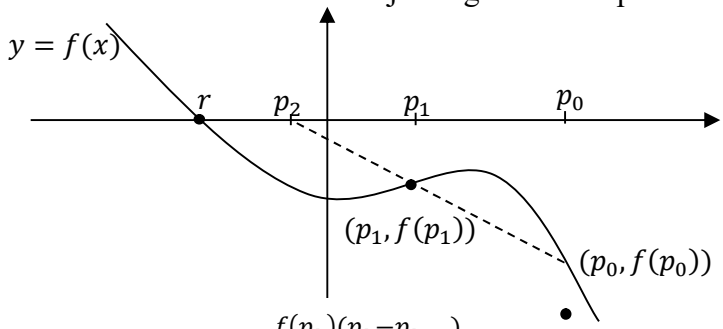
This means local maximum (or minimum) is in the interval. Sometimes the slope of $f'(p_0)$is small and the tangent line to the curve $y = f(x)$ is nearly horizontal. Then the sequence $\{p_k\}$ converges to some other root. Another possibility is cycling which occurs when the terms in the sequence $\{p_k\}$ tend to repeat.



Sometime the sequence does not converge for such a function $f(x) = \tan x$.

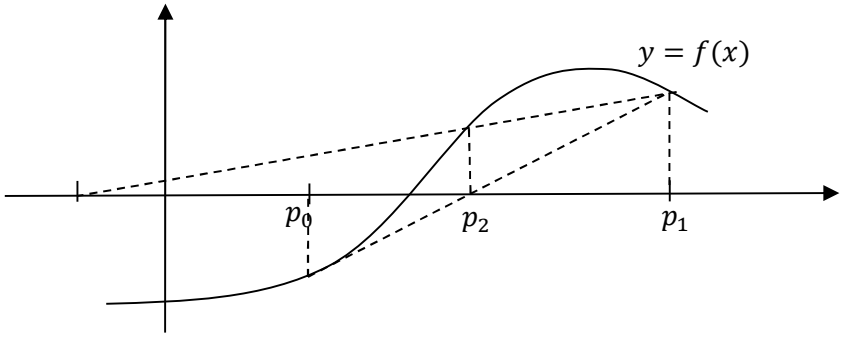


In **Secant Method**, two initial points $(p_0, f(p_0))$and $(p_1, f(p_1))$ near the root are needed. It is used the line joining these two points.

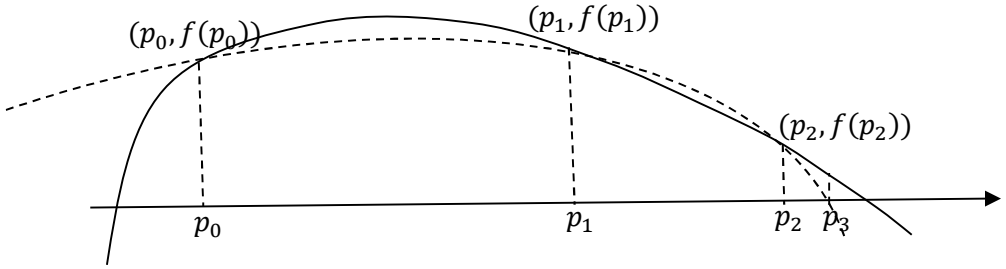


The formula is $p_{k+1} = p_k - \frac{f(p_k)(p_k - p_{k-1})}{f(p_k) - f(p_{k-1})}$for all k.

The convergence of this method is super linear. It is faster than a linear rate. It is almost as fast as Newton's Method. Secant Method needs only one function evaluation and is often faster in time, even though more iterates are needed to attain a similar accuracy with Newton's Method. The disadvantage of the secant method is sometime it may not converge when $f(p_k) \approx f(p_{k-1})$.



In **Muller's Method**, the three initial points $(p_0, f(p_0))$, $(p_1, f(p_1))$ and$(p_2, f(p_2))$ are needed. Theyare used to construct a parabola,second order polynomial, which is used to fit to the last three obtained points.



In this method, it is based on the variable $t = x - p_2$ and use the differences

$$h_0 = p_0 - p_2 \text{and} h_1 = p_1 - p_2.$$

And then use the quadratic polynomial, involving the variable,

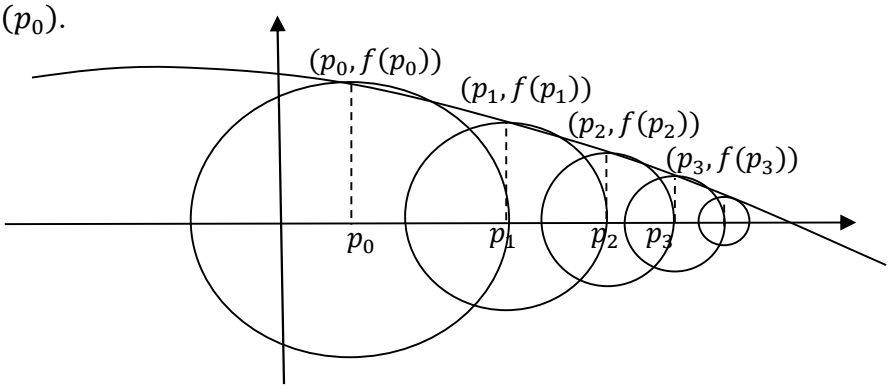
$$y = at^2 + bt + c\ .$$

This method can be used to find the imaginary roots and it is no needed to use derivatives. The convergence rate is faster than the Secant Method and almost as fast as Newton's Method.

The weakness of this method is three initial points are needed and extraneous roots can be found as this method used the quadratic formula.

New Method (Ohnmar Nwe's Method)

If $f(x)$ is continuous on the interval $[a, b]$ and it will across the X-axis at a root p that lie in the interval $[a, b]$. We draw a line by connecting the points $(p_0, 0)$and $(p_0, f(p_0))$. Then we construct a circle as centre $(p_0, 0)$ and radius $f(p_0)$.



Then this circle pass through the X-axis at $(p_1, 0)$. For the next step, construct the circle as centre $(p_1, 0)$ and radius $f(p_1)$. By proceeding, the centers of the circles are closer and closer to the root p. Here the equation of the circle with centre $(p_0, 0)$ and radius $f(p_0)$ is

$$(x - p_0)^2 + (y - 0)^2 = f(p_0)^2 \qquad (1)$$

This circle equation (1) pass through the X-axis at p_1. So that an equation relating p_1 and p_0 can be found.

$$(p_1 - p_0)^2 + (0 - 0)^2 = f(p_0)^2$$

$$p_1 = p_0 \mp f(p_0), \qquad (2)$$

When p_{k-1} and p_k are used in place of p_0 and p_1 the general rule is established as follow

$$p_k = p_{k-1} \mp f(p_{k-1}).$$

The convergence of this method is based on the idea that $|p_k - p_{k-1}| \to 0$ as $f(p_k) \to 0$.

The decision step for sign$(+)$ or $(-)$ is to analyze

$$|f(p_k)| < |f(p_{k-1})|.$$

Theorem (Ohnmar Nwe's Theorem)

Assume that $f \in C^1[a,b]$ and there exist a number $p \in [a,b]$ where $f(p) = 0$. Then there exist a $\delta > 0$ such that the sequence $\{p_k\}_{k=0}^{\infty}$ defined by the iteration

$$p_k = g(p_{k-1}) = p_{k-1} \mp f(p_{k-1}) \text{ for } k = 1,2,3, \ldots$$

will converge to p for any initial approximation $p_0 \in [p - \delta, p + \delta]$. (Here the function $g(x)$ is defined by $g(x) = x \mp f(x)$and it is used as the iteration function.)

Proof. We will use the fixed-point theorem to prove. We have to remind the fixed-point theorem.

(Fixed-point Theorem – Assume that $g(x)$ and $g'(x)$ are continuous on a balanced interval $(a,b) = (p - \delta, p + \delta)$ that contains the unique fixed point p and that the starting value p_0 is chosen in this interval.

If $|g'(x)| \leq K < 1$ for all $a \leq x \leq b$. Then the iteration $p_n = g(p_{n-1})$ will converge to p. In this case p is an attractive fixed-point.

If $|g'(x)| > 1$ for all $a \leq x \leq b$, then the iteration exhibits local divergence.)

In our method, the iteration function is

$$g(x) = x \mp f(x)$$

then $$g'(x) = 1 \mp f'(x)\ .$$

To be convergent, $g'(x)$ must be less than 1.

$$|g'(x)| = |1 \mp f'(x)| < 1$$

Therefore, a sufficient condition for p_0 to initialize a convergent sequence $\{p_k\}_{k=0}^{\infty}$ which converges to a root of $f(x) = 0$ is that $p_0 \in [p - \delta, p + \delta]$ and that δ be chosen so that

$$|1 \mp f'(x)| < 1 \quad \text{for all} \quad x \in [p - \delta, p + \delta].$$

Comparison of Newton's Method And Ohnmar Nwe's Method

Now we would like to express some examples that will show the comparison of convergent rate between Newton's method and our method.

Table 1. Comparison of convergent rate for the function $f(x) = x^3 + 3x + 2$ with $p_0 = 0$

k	Newton's Method	Ohnmar Nwe's Method
0	0	0
1	-0.6667	-2
2	-0.9333	
3	-0.9961	
4	-1.0000	

Table 2. Comparison of convergent rate for the function $f(x) = x^3 + 3x + 2$ with $p_0 = -2.5$

k	Newton's Method	Ohnmar Nwe's Method
0	-2.5	-2.5
1	-2.1250	-1.75
2	-2.0125	-1.9375
3	-2.0002	-1.9961
4	-2.0000	-2

Table 3. Comparison of convergent rate for the function $f(x) = x^3 + 3x + 2$ with $p_0 = -3$

k	Newton's Method	OhnmarNwe's Method
0	-3	-3
1	-2.3333	-1
2	-2.0667	
3	-2.0039	
4	-2.0000	

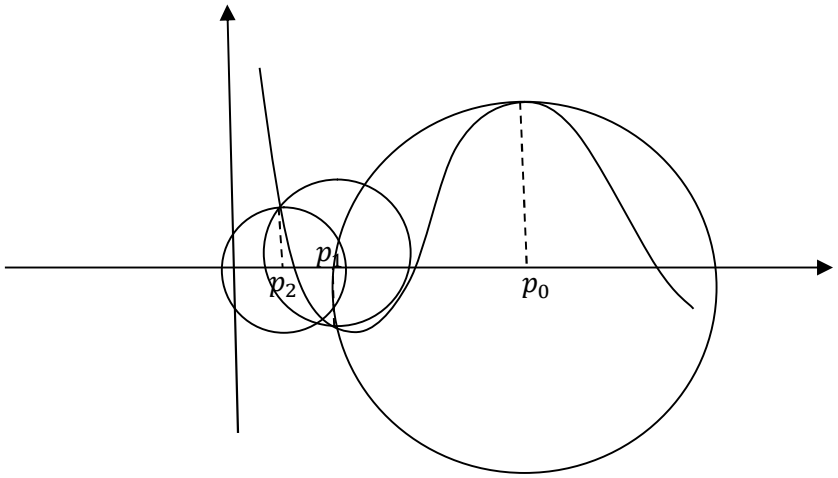
Table 4. Comparison of convergent rate for the function $f(x) = x^3 - 3x + 2$ with $p_0 = 0$

k	Newton's Method	OhnmarNwe's Method
0	0	0
1	0.6667	-2
2	0.8444	
3	0.9244	
4	0.9627	
5	0.9815	
6	0.9908	
7	0.9954	
8	0.9977	
9	0.9988	
10	0.9994	
11	0.9997	
12	0.9999	
13	0.9999	
14	1.0000	

Our method is needed only one initial point and convergent rate is faster than the Newton's method if the function has enough slope. The formula is also very simple and it is very easy to calculate.

Pitfall of Ohnmar Nwe's Method

There are some difficulties to use our method if the function $f(x)$ has several roots in the interval that we consider. At that condition, it may be jump of some roots although it gives a root . Another possibility is out of the interval if $f(p_0)$ is very large.



Acknowledgement

This work was supported from the project fund of "Finding a New Method for the Solution of Nonlinear Equations $f(x) = 0$ (2017-2018)", Mathematics Department, Loikaw University.

References

1. Mathews, J. H.,(1992) *Numerical Methods for Mathematics, Science, and Engineering*, Second edition, Prentice-Hall International Inc., U.S.A.
2. Lambers, J., "The Secant Method", www.doumbase.com> Secant-Method-…

WAVELETS ON GROUPS

KhinWah Win*

Abstract

This paper is an expository survey of basic concepts of Wavelet Analysis. We shall discuss wavelets in L^2 (R) and L^2 (G), where G is a locally compact abelian Group, in particular a Lie group. A discussion of basic facts of Topological groups, Differentiable Manifolds and Lie groups are also included.

Keywords: Haar measure, Topological group, Differentiable Manifold, Lie Group, Wavelets.

1. Introduction

The classical Fourier theory is concerned with the study of the Fourier transform $\hat{f}$ of a given function f:

(1) $$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-ix\xi}dx$$

and its inversion problem i.e. studying conditions under which the following inversion relation holds:

(2) $$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)e^{i\xi x}dx.$$

The corresponding Foureir sereis theory is the investigation of the validity of the relation.

(3) $$f(x) = \sum_{n\in Z} \hat{f}(n)e^{inx}$$

in various modes of convergence, where

(4) $$\hat{f}(n) = \int_{-\infty}^{\infty} f(x)e^{-inx}dx.$$

The infinite series

(5) $\sum_{n\in Z} \hat{f}(x)e^{inx}$ is called the Fourier series of f and the numbers $\hat{f}(x)$ are called the fourier coefficients of f. The Euler relation, $e^{ix} = \cos x + i\sin x$,

*. Lecturer, Department of Mathematics, University of Yangon

shows that the Fourier sereis in (5) is infact a series in sine and cosine. In application, we usually have to approximate a function by its Fourier series,

$$f(x) \sim \sum_{n=-\infty}^{\infty} \hat{f}(x) e^{inx}$$

and thus have to compute the infinite integral

$$\hat{f}(n) = \int_{-\infty}^{\infty} f(x) e^{-inx} dx.$$

Except for very nice functions, this integral cannot be evaluated in closed form.

Again we have to approximate this infinite integral on some suitable finite interval. It is therefore desirable that the integrand decays at infinity.

The problem here is that sine and cosine functions do not decay at infinity i.e., $|\sin x|, |\cos x| \not\longrightarrow 0$ as $|x| \to \infty$. They remain oscillatory on the whole real line.

Wavelets are an attempt to replace sine and cosine functions with function having sufficient rate of decay at infinity.

Wavelets (on *P*) were introduced in early nineteen eighties by Morlet, Arens, Fourgeau and Giard. Later the mathematical foundations of the wavelet theory was laid down by I. Daubechies and Y. Meyers [7]. This paper is an exposition of the extension of wavelet theory from *R* to topological groups.

2. Fourier Analysis on Groups

In this section we briefly discuss Fourier Analysis on Groups. For details, we refer to [8].

2.1 Haar Measure

2.1.1Definition. A **topological group** *G* is a group which is also a topological space such that the group operations

$$G \times G \quad \to \quad G$$

$$(x, y) \quad \mapsto \quad xy,$$

and

$$G \quad \to \quad G$$
$$x \quad \mapsto \quad x^{-1}$$

are continuous.

A topological group G is locally compact if it is locally compact as a topological space.

2.1.2 Proposition. Let G be a locally compact abelian group (LCA). Then there exists a non-negative regular and translation invariant Borel measure on G. This measure is called the ***Haar measure*** on G. [8].

Some function spaces of interest on G are as follows:

- $C(G)$ denotes the set of all continuous complex functions on G.
- $C_C(G)$ denote the set of all continuous complex functions on G with compact support.
- $L^P(G), 1 \le p < \infty$ the set of all Borelmeasurable functions on G such that

$$\left(\int_G | f |^P \, dx \right)^{1/p} < \infty$$

where dx is the Haar measure on G.

2.2 The Dual Group and the Pontryagin Duality

2.2.1 Definition. Let G be a LCA group. A complex function γon G is called a ***character*** of G if for all $x, y \in G$,

(i) $| \gamma(x) | = 1$

(ii) $\gamma(xy) = \gamma(x)\gamma(y)$.

2.2.2 Remark.

(1) γ is a homomorphism of the group G and the multiplicative group S^1 of the unit circle in X.

(2) The example of a character is the exponential map

$$R \to S^1$$
$$x \mapsto e^{ix}.$$

2.2.3 Definition. Let Γ be the set of all continuous characters of G. Γ is a group with respect to the addition defined by

$$(\gamma_1 + \gamma_2)(x) = \gamma_1(x)\gamma_2(x), \qquad (\gamma_1, \gamma_2 \in \Gamma, x \in G)$$

$(\Gamma, +)$ is called the ***dual group of G***.

It is customary to write the "duality notation" (x, γ) for $\gamma(x)$.

2.2.4 Theorem (The Pontryagin duality)

Let G be a LCA group and Γbe its dual group. Let $\hat{\Gamma}$ be the dual group of Γ. Then $\hat{\Gamma} \simeq G$. [8]

2.3 The Fourier Transform on Locally Compact Abelian Group

Let G be a LCA group and Γbe its dual group. Let $f \in L^1(G)$. A function $\hat{f}$ defined on Γ by

$$\hat{f}(\gamma) = \int_G f(x)(-x, \gamma)dx, (\gamma \in \Gamma)$$

is called the **Fourier transform** of f.

This generalization of the ***classical*** Fourier transform on R to a LCA group G is only too natural, since R is also a LCA group.

As may be expected the following classical results still hold:

(1) $\int_G f(x)g(x)dx = \int_\Gamma \hat{f}(\gamma)\hat{g}(\gamma)d\gamma,$ (Parseval)

(2) $\| f \|_2 = \| \hat{f} \|_2,$ (Plancherel) [8]

2.4 Lie groups

In doing Fourier Analysis especially wavelet Analysis, it is sometimes necessary to consider smooth functions also.

For this purpose we have to consider topological spaces which have also differentiable structure.

A **differentiable manifold** is a Hausdorff topological space X such that each point in X has a neighbourhood homeomorphic to an open set in R^n.

2.4.1 Definition. A topological group G which is also a differentiable manifold such that the maps

$$\begin{array}{rcl} G \times G & \to & G \\ (x, y) & \mapsto & xy \qquad \text{and} \\ G & \to & G \\ x & \mapsto & x^{-1} \end{array}$$

are smooth is called a **Lie group**.

For our purpose the following matrix Lie groups will suffice.

Let k be the real field R or the complex field X.

Let $M_n(k)$ be the set of all $n \times n$ matrices with entries from k.

(1) The general linear group $GL_n(k) = \{A \in M_n(k) : \det A \neq 0\}$.

(2) The special linear group $SL_n(k) = \{A \in GL_n(k) : \det A = 1\}$.

(3) The orthogonal group $O_n(k) = \{A \in GL_n(k) : \overline{A^T} A = I_n\}$.

Clearly $GL_n(k), SL_n(k)$ and $O_n(k)$ are groups with respect to ordinary matrix multiplication.

Each $n \times n$ matrix A can be identified as a point of k^{n^2}.

So $GL_n(k)$, $SL_n(k)$ and $O_n(k)$ are also topological groups with the topology induced by k^{n^2}. For details we refer to [1].

3. Wavelets

3.1 Wavelets on *P*

3.1.1 Definition. Let $\psi \in L^2(R)$ with sufficient rate of decay at infinity.

Consider the family

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k),\ (j,k \in Z)$$

of translations and dialations of ψ .

Suppose that $\{\psi_{j,k}\}$ forms an orthonormal basis for $L^2(R)$; i.e.,

$$\begin{aligned} \langle \psi_{j,k}, \psi_{l,m} \rangle &= \delta_{j,l}\delta_{k,m} \\ f &= \sum_{j,k \in Z} \langle f, \psi_{j,k} \rangle \psi_{j,k}. \end{aligned}$$

Then ψ is called a mother wavelet and the system $\{\psi_{j,k}\}$ is called a wavelet basis.

Example (1). The Haar function is a wavelet with compact support.

$$\psi(x) = \begin{cases} 1 & \text{if} \quad 0 \le x < \dfrac{1}{2}, \\ -1 & \text{if} \quad \dfrac{1}{2} \le x < 1, \\ 0 & \text{elsewhere} \end{cases}$$

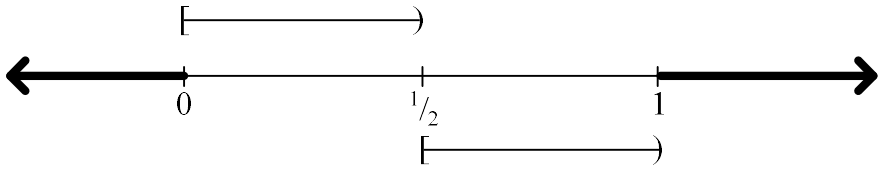


Figure (1)

Example (2). The Gaussian function is a smooth wavelet with fast rate of decay at infinity.

$$\psi(x) = e^{-x^2}$$

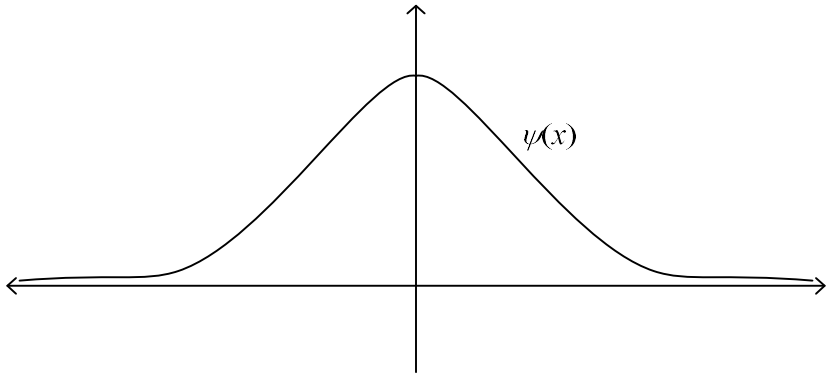


Figure (2)

3.2 Group theoretical approach to wavelets.

We recall some concepts of group representations [9].

3.2.1 Definition. Let G be a locally compact group. A **unitary representation** of G is a pair (π, H) where H is a Hilbert space and π is a continuous homomorphism of G into the group $U(H)$ of unitary operators on H that is the operations are continuous.

$$\pi(xy) = \pi(x)\pi(y),$$

$$\pi(x^{-1}) = (\pi(x))^{-1} = (\pi(x))^{*} \quad \text{for } x, y \in G.$$

3.2.2The *Affine* Group

Let $G = \{(a,b) \in R^{\times} \times R, a \neq 0\}$.

Define the operation $(a,b)\cdot(c,d) = (ac, d + \frac{b}{c})$.

Then G is a group called the "***Affine*** group" with $(a,b)^{-1} = (a^{-1}, -ab)$.

For $g = (a,b) \in G$, define

$$(T_g\psi)(x) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{x-b}{a}\right) = \frac{1}{\sqrt{|a|}}\psi(g^{-1}(x))$$

where $\psi \in L^2(R)$.

Then the mapping,

$$T : G \to U(L^2(R))$$

$g \mapsto T_g$ is a unitary representation of G on $L^2(R)$, [11].

3.2.3 The continuous wavelet transform

Let $\psi \in L^2(R)$ be a wavelet. Then the mapping $W_\psi : L^2(R) \to L^2(R^\times \times R)$ defined by

$$\begin{aligned}(W_\psi f)(g) &= \int_R f(x)\overline{(T_g\psi)(x)}dx \\ &= \langle f, T_g\psi \rangle \\ &= \langle f, \psi_{a,b} \rangle\end{aligned}$$

is called the continuous wavelet transform.

The main purpose of wavelet Analysis is [like that of Fourier Analysis] to look for conditions such that the **Calderon**reproducing formula

$$\int_G (W_\psi f)(g)\psi(x)d\lambda(g) = f$$

holds, where $\lambda$is the Haar measure on G [11].

References

1. Baker. *A*. An Introduction to matrix groups and their applications. *Dept. of Math. Uni. Glasgow.* (2000).
2. Benedetto, JJ, Generalized HA and Gabor and Wavelet Systems. *Proceedings of Symposia in Applied Mathematics,* Volume 00, 19xx.
3. Daubechies, I, Orthonormal Bases of Compactly supported wavelets. II. Variations on a theme. *SIAM J. MATH. ANAL* Vol. 24, No. 2, JP 499-519, March (1993).
4. Daubechies. I, Orthonormal Bases of Compactly supported wavelets. *Commun Pure and Applied Mathematics.*Vol XL1 909-996 (1988).
5. Heil, C. E, Walnut, D. F., Continuous and discrete wavelet transforms. *SIAM Review* Vol.31, No.4, pp. 628-666. December (1989).
6. Mallet, S. C., A Theory for Multiresolution signal Decomposition: The Wavelet Representation IEE Transaction Pattern Analysis and Mechanic Intelligence. Vol 11. No. 7 July (1989).
7. Meyer. Y., *Wavelets and Operators,* Cambridge University Press, (1992).
8. Rudin, W. *Fourier Analysis on Groups,* Interscience, (1967).
9. Steinberg, S., *Representation Theory of Finite Groups,* Caleton Uni. (2009).
10. Walter, J., Fourier Analysis and Wavelet Analysis. *Notices of the AMS* Vol. 44, No.6.
11. Welss, G., Wilson, E. N., Mathematical Theory of Wavelets. Dept. Math. Washington University.

A STUDY ON FILTERS

Khin Moe Moe*

Abstract

In this paper, firstly the classification of filters is discussed. Topological space plays a crucial role in this discussion. After that the relations between them are studied. Especially, the characterizations of ultrafilters are presented with detail proofs. Ultrafilter is a powerful tool both in set theory and in topology. Moreover, the comparisons of filters are expressed and some notions of filter basis and trace of filter are described. Finally, ultrafilter convergence theorem and convergence of Cauchy filter in topological vector space are investigated.

1. Some types of filters

1.1 Definitions

A collection $\mathcal{F}$ of subsets of a set X is called a **filter** on X if it satisfies the following axioms:

(F1) If A$\subset$ X and A contains a set B $\in \mathcal{F}$, then A $\in \mathcal{F}$.

(F2) The intersection of a finite collection of sets in $\mathcal{F}$ belongs to $\mathcal{F}$.

(F3) The empty subset of X does not belong to $\mathcal{F}$.

First let us examine a few elementary consequences of this definition. It follows from (F1) that X is a member of any filter on X.

Note that P(X) a collection of subsets of X is not a filter on X. However, it satisfies (F1) and (F2). Therefore it is sometimes called the improper filter on X. Conversely, if $\mathcal{F}$ is a collection of subsets of X containing the empty set and satisfying (F1) and (F2), it follows from (F1) that $\mathcal{F} =$ P(X), that is, $\mathcal{F}$ is improper.

Let X be a set and $\mathcal{A} \subset$ P(X) a collection of subsets. Then $\mathcal{A}$ has the **finite intersection property** (FIP) if any finite intersection of sets in$\mathcal{A}$ is non-empty. (From the axioms (F2) and (F3) that a filter has the FIP.)

The **cofinite filter** on an infinite set X is the set of all subsets A of X such that the complement of A in X is finite.

*. Dr, Lecturer, Department of Mathematics, Monywa University

That is, $\mathcal{F} = \{A \subseteq X: X \setminus A \text{ is finite }\}$. (or)

$$\mathcal{F} = \{X \setminus A : A \subseteq X \text{ is finite }\}.$$

This filter on $X = \mathbb{N}$, the set of natural numbers, is also called the **Fréchet filter**on $\mathbb{N}$.

A maximal element of the set of all filters on X is called an **ultrafilter** on X.

For any non-empty subset M of X, the set $\{A \subseteq X : M \subseteq A\}$ is a filter on X, the **principal filter** generated by M.

For any $a \in X$ the set $\{ A \subseteq X : a \in A \}$ is the **principal ultrafilter** defined by a.

Any ultrafilter that is not principal is called **non-principal ultrafilter**.

A filter $\mathcal{F}$ on X is**free** if the intersection of all sets in $\mathcal{F}$ is empty. That is, $\cap_{A \in \mathcal{F}} A = \emptyset$.

Let X be a set and $\mathcal{A} \subset P(X)$ a collection of subsets. The (im)proper filter generated by $\mathcal{A}$ is the set

$<\mathcal{A}> = \cap(\{ \mathcal{F} \subset P(X) : \mathcal{F} \supset \mathcal{A}$ and $\mathcal{F}$ is a(n) (im) proper filter on X$\})$.

So $<\mathcal{A}>$ is the intersection of all (im) proper filters on X that contains set $\mathcal{A}$.

1.2 Example

The set of all neighborhoods of a point $x \in X$ is filter $\mathcal{B}(x)$ called the neighborhood filter of x.

2.Characterizations of the Ultrafilters

2.1 Theorem (Ultrafilter lemma)

Let X be a set and suppose $\mathcal{A} \subset P(X)$ has FIP. Then there is an ultrafilter $\mathcal{U}$ on X which contains all of $\mathcal{A}$.

Proof

Let the set $\mathfrak{V}$ consisting of all proper filters on X containing $\mathcal{A}$, partially ordered by set inclusion. Then $\mathfrak{V}$ is non-empty because $<\mathcal{A}>\in \mathfrak{V}$. Let $\mathcal{C}$ be a chain in $\mathfrak{V}$.

We will prove $\cup \mathcal{C} \in \mathfrak{V}$. For (F3), since any element of $\mathcal{C}$ does not contain empty set, $\emptyset \notin \cup \mathcal{C}$.For (F2), if A, B$\in \cup \mathcal{C}$, then there are C, D $\in \mathcal{C}$ such that A$\in$ C and B$\in$ D. Since $\mathcal{C}$ is a chain, we have C $\subset D$ without loss of generality. Consequently, A, B are elements of D and since D is a filter A$\cap$ B $\in$ D$\in \cup\ \mathcal{C}$. For (F1), it is a trivial matter to verify that $\cup \mathcal{C}$ is closed under supersets, so we have $\cup \mathcal{C} \in \mathfrak{V}$ indeed. This union is an upper bound of $\mathcal{C}$ in $\mathfrak{V}$. According to Zorn's lemma, $\mathfrak{V}$ has maximal elements. Let $\mathcal{U}$ be a maximal element of $\mathfrak{V}$. If $\mathcal{F} \supset \mathcal{U}$ is a filter, then $\mathcal{A} \subset \mathcal{F}$.By the maximality of $\mathcal{U}$, $\mathcal{F} \subset \mathcal{U}$ and we have $\mathcal{U} = \mathcal{F}$.So $\mathcal{U}$ is an ultrafilter and it contains all of $\mathcal{A}$.

2.2 Lemma

Let $A_1, A_2, \ldots, A_n \in P(X)$ such that $A_1 \cup A_2 \cup \ldots \cup A_n \in \mathcal{U}$ where$\mathcal{U}$ is an ultrafilter on X. Then $A_i \in \mathcal{U}$ for at least one i. In addition, if the sets are mutually disjoint, then $A_i \in \mathcal{U}$ for exactly one i.

Proof

Let $A_1 \cup A_2 \in \mathcal{U}$. Suppose (to the contrary) that neither $A_1 \in \mathcal{U}$ nor $A_2 \in \mathcal{U}$. Consider $\mathcal{M} = \{Z \in P(X) : A_1 \cup Z \in \mathcal{U}\}$.

First we need to show that $\mathcal{M}$ is a filter on X.For (F3), if $\emptyset$ is a member of $\mathcal{M}$, then $A_1 = A_1 \cup \emptyset \in \mathcal{U}$, contradiction. So $\emptyset \notin \mathcal{M}$.

For (F2), if $B_1, B_2 \in \mathcal{M}$, then $A_1 \cup B_1 \in \mathcal{U}$ and $A_1 \cup B_2 \in \mathcal{U}$. Now $(A_1 \cup B_1) \cap (A_1 \cup B_2) \in \mathcal{U}$ because $\mathcal{U}$ is a filter.That is, $A_1 \cup (B_1 \cap B_2) \in \mathcal{U}$.It follows that$B_1 \cap B_2 \in \mathcal{M}$.

For (F1), let V$\in$ P(X), U $\subset$ V and U $\in \mathcal{M}$. Then $A_1 \cup U \in \mathcal{U}$. Since U $\subset$ V, $A_1 \cup U \subset A_1 \cup V$.Thus $A_1 \cup V \in \mathcal{U}$ because $\mathcal{U}$ is a filter. So V $\in \mathcal{M}$.

Therefore $\mathcal{M}$ is a filter on X.Moreover, we have $\mathcal{U} \subseteq \mathcal{M}$. Also, $\mathcal{U} \subsetneq \mathcal{M}$because $A_2 \in \mathcal{M} \backslash \mathcal{U}$, contradicting the maximality of $\mathcal{U}$.Our assumption is false, so $A_i \in \mathcal{U}$ for at least one i.Finally, if $A_1 \cap A_2 = \emptyset$ and $A_1, A_2 \in \mathcal{U}$ then this implies that $\emptyset \in \mathcal{U}$, a contradiction.The generalization to n≥2 follows by induction.

2.3 Theorem

Let $\mathcal{F}$ be a filter on X. Then $\mathcal{F}$ is an ultrafilter if and only if for every A ⊂ X either A ∈ $\mathcal{F}$or X \ A ∈ $\mathcal{F}$.

Proof

Suppose $\mathcal{F}$ is an ultrafilter.Let A ∈ P(X).The previous lemma holds since A ∪ (X \ A) = X ∈ $\mathcal{F}$ and A ∩(X \ A) = ∅.

Conversely, suppose (to the contrary)that $\mathcal{F}$ is not an ultrafilter. Then there exists a filter $\mathcal{M}$ such that $\mathcal{F} \subsetneq \mathcal{M}$ and take A ∈ $\mathcal{M} \setminus \mathcal{F}$.

Thus A ∈ $\mathcal{M}$ and A∉ $\mathcal{F}$. So X \ A ∈ $\mathcal{F}$ by given condition.

Since $\mathcal{F} \subset \mathcal{M}$, then this implies that both A and X \ A are in $\mathcal{M}$. Hence A ∩(X \ A) = ∅ ∈ $\mathcal{M}$, contradicting the fact that $\mathcal{M}$ is a filter.

2.4 Remark

If $\mathcal{U}$ is an ultrafilter on X, and A∈ $\mathcal{U}$, then $\mathcal{U}$ contains all sets B with A ⊂ B ⊂ X. Indeed, if we start with such a B, then by the above result, either B ∈ $\mathcal{U}$ or X \ B ∈ $\mathcal{U}$.If X \ B ∈ $\mathcal{U}$, then A ∩ (X \ B) = ∅ ∈ $\mathcal{U}$, contradiction. Therefore B must belong to $\mathcal{U}$.

2.5 Corollary

The Fréchet filter $\mathcal{F}$on an infinite set$\mathbb{N}$ is not an ultrafilter.

Proof

Let $\mathbb{E}$ and $\mathbb{O}$ denote the sets of the even and odd numbers in $\mathbb{N}$ respectively.We known that $\mathbb{E} \cap \mathbb{O} = \emptyset$ and$\mathbb{E} \cup \mathbb{O} = \mathbb{N} \in \mathcal{F}$, but neither $\mathbb{E}$ nor $\mathbb{O}$ belongs to $\mathcal{F}$ because any set in $\mathcal{F}$has finite complement.

3. Types of Ultrafilters

There are two very different types of ultrafilters such as principal and non-principal (free).

3.1 Proposition

Any ultrafilter over a finite set is principal.

Proof

Let X be a finite set, $\mathcal{U}$ be an ultrafilter over P(X) and

$\mathcal{U} = \{S_1, S_2, \ldots, S_k\}$.Since $\emptyset \notin \mathcal{U}$ and $S_i \cap S_j \in \mathcal{U}$ for every i, j,

$S_1 \cap S_2 \cap \ldots \cap S_k \neq \emptyset$. If a $\in \cap_{i=1}^{k} S_i$, then a $\in \mathcal{U}$.But by the definition of principal ultrafilter$\{ S : a \in S\} \subset \mathcal{U}$.By the maximality of ultrafilter, $\mathcal{U} = \{S: a \in S\}$.

(or)

Let A be a finite set. Then either some a ∈ A satisfies {a} is in the ultrafilter, in which case it is principal; or else X \ {a} is in the ultrafilter for all a∈ A, so the finite intersection

$$A \cap(\cap_{a\in A} (X \setminus \{a\}) = A \cap(X \setminus \cup_{a\in A} \{a\}) = A \cap (X \setminus A) = \emptyset$$

is also in the ultrafilter.

So a non-principal ultrafilter must contain only infinite sets. In particular, if X is finite, then every ultrafilter on X is principal.

3.2 Proposition

Cofinite filter is intersection of all non-principal ultrafilters.

Proof

Let X be an infinite set.

Suppose that a set A⊆ X; we want to show that A is cofinite. Suppose for contradiction that A is not cofinite. That is, the set D= X \ A is infinite. From Proposition 3.1, the infinite set D belongs to some non-principal ultrafilter $\mathcal{U}$ on X. But $\mathcal{U}$ is a non-principal ultrafilter on X which does not

contain A, contradicting our assumption that A belongs to every non-principal ultrafilter.

Let $\mathcal{F} = \{B \subseteq X: X \setminus B \text{ is finite}\}$, the cofinite filter on X. Then the collection $\{D\} \cup \mathcal{F}$ has the FIP, whence $\{D\} \cup \mathcal{F} \subset \mathcal{U}$ for some ultrafilter $\mathcal{U}$. Since $\mathcal{U}$ contains $\mathcal{F}$, it is non-principal.

3.3 Corollary

A non-principal ultrafilter is free.

Proof

If there exists $x \in \bigcap_{A \in \mathcal{F}} A$, then $X \setminus \{x\}$ is not an element of $\mathcal{F}$, by Theorem 2.3, $\{x\} \in \mathcal{F}$ and $\mathcal{F}$ is a principal ultrafilter.

3.4 Proposition

Every non-principal ultrafilter on an infinite set X contains the cofinite filter on X.

Proof

Let $\mathcal{U}$ be a non-principal ultrafilter on X and let $x \in X$ be arbitrary. Since $\mathcal{U}$ is an ultrafilter, exactly one of the sets $\{x\}$ and $X \setminus \{x\}$ belongs to $\mathcal{U}$, and since $\mathcal{U}$ is non-principal, $\{x\} \notin \mathcal{U}$. Thus, $X \setminus \{x\} \in \mathcal{U}$ for each $x \in X$. Now let F be any finite subset of X; then

$$X \setminus F = X \setminus \bigcup_{x \in F} \{x\} = \bigcap_{x \in \mathcal{F}} (X \setminus \{x\}) \in \mathcal{U}.$$

That is, $X \setminus F \in \mathcal{U}$. We have $\{X \setminus F: F \subseteq X \text{ is finite}\}$ is the cofinite filter on X.

Therefore $\mathcal{U}$ contains the cofinite filter.

3.5 Proposition

An ultrafilter on X is free if and only if it contains the Fréchet filter on X.

Proof

In the previous proposition we proved that every free ultrafilter on X contains the Fréchet filter on X. For the converse, suppose that $\mathcal{U}$ is a fixed (principal) ultrafilter on X; then there is an x∈ X such that {x} ∈ $\mathcal{U}$.

But X \ {x} is an element of the Fréchet filter that is not in $\mathcal{U}$, so $\mathcal{U}$ does not contain the Fre'chet filter.

4. Ultrafilter Convergence Theorem

4.1 Definition

A filter $\mathcal{F}$ on a topological space Y **converges** to a point y ∈ Y or y is a **limit** of $\mathcal{F}$ if for all open sets U containing y, U ∈ $\mathcal{F}$.

4.2 Theorem

Let Y be a topological space.

1. Y is Hausdorff if and only if every ultrafilter $\mathcal{F}$ on Y converges to at most one point.
2. Y is compact if and only if every ultrafilter $\mathcal{F}$ on Y converges to at least one point.

Proof

1. Suppose (to the contrary) that Y is Hausdorff, but x ≠ y are limit points of $\mathcal{F}$.

Since Y is Hausdorff, there exist disjoint open sets x ∈ U and y ∈V .By the definition of limit point, U, V ∈ $\mathcal{F}$ but U ∩ V= ∅, contradiction.

Conversely, suppose that Y is not Hausdorff. Then there are points x ≠ y such that every open neighborhood of x intersects every open neighborhood of y.

Then {U : x ∈ U open} ∪ { V : y ∈ V open } has the FIP. Let $\mathcal{F}$ be an ultrafilter containing it. So x and y are both limit points of $\mathcal{F}$.

2. Suppose to the contrary that Y is compact, but $\mathcal{F}$ has no limit points. Then for all y ∈ Y, there is an open set $U_y$containing y such that $U_y \notin \mathcal{F}$. So $Y = \cup_{y \in Y} U_y$ and by compactness, $Y = \cup_{i=1}^{n} U_{y_i}$. But Y∈ $\mathcal{F}$, so some $U_{y_i} \in \mathcal{F}$, contradiction.

Conversely, suppose that Y is not compact. Then there is an open cover $Y = \cup_{y \in Y} U_y$with no finite subcover. So $\cap_i (Y \setminus U_i) = \emptyset$, but no finite intersection is empty. Then $\{(Y \setminus U_i)\}_i$has the FIP, so we can take an ultrafilter $\mathcal{F}$ containing it. Now for any point y ∈ Y, y is contained in some U_i, and $U_i \notin \mathcal{F}$, since$(Y \setminus U_i) \in \mathcal{F}$. So y is not a limit point of $\mathcal{F}$.

5.Comparison of filters on a set X

5.1 Definition

Let $\mathcal{F}_1$, $\mathcal{F}_2$ be two filters defined on a set X. We say that $\mathcal{F}_1$is **finer** than $\mathcal{F}_2$(or that $\mathcal{F}_2$is **coarser** than $\mathcal{F}_1$) if $\mathcal{F}_2 \subset \mathcal{F}_1$.

5.2 Proposition

Let $(\mathcal{F}_i)_{i \in I}$ be a family of filters on a set X. Then $\mathcal{F} = \cap_{i \in I} \mathcal{F}_i$ is a filter on X and has the following properties.

(a) $\mathcal{F}$ is coarser than $\mathcal{F}_i$(i ∈ I).

(b) If $\mathcal{F}'$is a filter coarser than every$\mathcal{F}_i$(i ∈ I) then$\mathcal{F}' \subset \mathcal{F}$.

Proof

For (F1), let A⊂X, B⊂A and B∈ $\mathcal{F}$. It follows that B ∈ $\mathcal{F}_i$ for every i∈ I.

Since $\mathcal{F}_i$ is a filter and B⊂A, A∈ $\mathcal{F}_i$ for every i ∈ I.

Thus A $\in \cap_{i \in I} \mathcal{F}_i = \mathcal{F}$.

For (F2), let $A_1, A_2, \ldots, A_n \in \mathcal{F}$. For each j, $A_j \in \cap_{i \in I} \mathcal{F}_i$ and $A_j \in \mathcal{F}_i$ for every i ∈ I.Since $\mathcal{F}_i$ is a filter (i ∈ I), $\cap_{j=1}^{n} A_j \in \mathcal{F}_i$(i ∈ I).So $\cap_{j=1}^{n} A_j \in \cap_{i \in I} \mathcal{F}_i = \mathcal{F}$.

For (F3), for each i, $\mathcal{F}_i$ does not contain empty set, $\emptyset \notin \cap_{i\in I} \mathcal{F}_i$. Therefore $\mathcal{F}$ is a filter. Since $\mathcal{F} = \cap_{i\in I} \mathcal{F}_i$, $\mathcal{F} \subset \mathcal{F}_i (i \in I)$.That is, $\mathcal{F}$ is coarser than $\mathcal{F}_i (i \in I)$.

(a) Let $\mathcal{F}'$ be a filter coarser that $\mathcal{F}_i (i \in I)$. That is, $\mathcal{F}' \subset \mathcal{F}_i (i \in I)$.
Thus $\mathcal{F}' \subset \cap_{i\in I} \mathcal{F}_i = \mathcal{F}$.

5.3 Definition

Let $(\mathcal{F}_i)_{i\in I}$ be a family of filters $\mathcal{F}_i$defined on set X. If there exists a filter $\mathcal{F}$ on X such that

(glb1)$\mathcal{F}$ is coarser than every $\mathcal{F}_i (i \in I)$.

(glb2) If $\mathcal{F}'$ is a filter on X such that $\mathcal{F}'$is coarser than every $\mathcal{F}_i (i \in I)$, then $\mathcal{F}' \subset \mathcal{F}$.Then $\mathcal{F}$ is called **the greatest lower bound** of the family $(\mathcal{F}_i)_{i\in I}$. Proposition (5.2) implies the greatest lower bound of a family $(\mathcal{F}_i)_{i\in I}$ of filters $\mathcal{F}_i$ on X always exists.

5.4 Definition

Let $(\mathcal{F}_i)_{i\in I}$ be a family of filters $\mathcal{F}_i$ on X. If there exists a filter $\bar{\mathcal{F}}$ on X such that

(lub1) $\bar{\mathcal{F}}$ is finer than every $\mathcal{F}_i (i \in I)$.

(lub2) If $\bar{\mathcal{F}}'$ is a filter on X such that $\bar{\mathcal{F}}'$is finer than every $\mathcal{F}_i, i \in I$, then $\bar{\mathcal{F}}$is called the **least upper bound** of $(\mathcal{F}_i)_{i\in I}$.

5.5 Proposition

Let $(\mathcal{F}_i)_{i\in I}$ be a family of filters on a set X. Then this family has a least upper bound in the set of all filters on X if and only if there exists a filter on X which is finer than every $\mathcal{F}_i$ for $i \in I$.

Proof (Necessary condition)

Assume that the least upper bound $\bar{\mathcal{F}}$ exists.(lub 1) implies $\bar{\mathcal{F}}$ is finer than every $\mathcal{F}_i$ for $i \in I$.

(Sufficient condition)

Assume that there exists a filter $\mathcal{F}$ on X which is finer than $\mathcal{F}_i$ (i ∈ I).Let Φ be the set of all filters which are finer than $\mathcal{F}_i$ (i ∈ I).Then $\mathcal{F} \in \Phi$ and so Φ is non-empty.Let $\bar{\mathcal{F}}$ be the greatest lower bound of Φ.We prove that $\bar{\mathcal{F}}$ is least upper bound of $\mathcal{F}_i$ (i ∈ I).

Let $\mathcal{F}_j \in (\mathcal{F}_i)_{i\in I}$. Since $\bar{\mathcal{F}}$ is the greatest lower bound of $\mathcal{F}_i$ (i ∈ I), $\mathcal{F}_j \subset \bar{\mathcal{F}}$.

That is, $\bar{\mathcal{F}}$ is finer than every $\mathcal{F}_i$ (i ∈ I).Put g ∈ Φ. Then g is finer than every $\mathcal{F}_i$.Thusevery $\mathcal{F}_i$ is coarser than g of Φ. Hence $\mathcal{F}_i \subset \bar{\mathcal{F}}$.

Let $\mathcal{F}'$ be a filter on X such that$\mathcal{F}'$ is finer than every $\mathcal{F}_i$ (i ∈ I).

Then $\mathcal{F}' \in \Phi$. Since $\bar{\mathcal{F}}$ be the greatest lower bound of Φ and $\mathcal{F}' \in \Phi$, (glb1) implies $\bar{\mathcal{F}} \subset \mathcal{F}'$.Therefore $\bar{\mathcal{F}}$ is the least upper bound of $(\mathcal{F}_i)_{i\in I}$.

6. Some Notions of Filter Basis and Trace of Filter

6.1 Definition

A collection $\mathfrak{B}$ of subsets of X is a **filter basis** if it satisfies the following two conditions:

(FB1) The intersection of two sets in $\mathfrak{B}$ contains a set of $\mathfrak{B}$.

(FB2) $\mathfrak{B}$ is non-empty and the empty subset of X does not belongs to $\mathfrak{B}$.

6.2 Definition

Let f: X⟶Y be a mapping from a set X into a set Y. Let $\mathfrak{B}$ be a filter basis on Y.Define $f^{-1}(\mathfrak{B}) = \{ f^{-1}(A) : A \in \mathfrak{B} \}$.

6.3 Proposition

Let $\mathfrak{B}$ be a filter basis on Y and f: X⟶Ybe a mapping. Then $f^{-1}(\mathfrak{B})$ is a filter basis on X if and only if $f^{-1}(A) \neq \emptyset$ for every A ∈ $\mathfrak{B}$.

Proof

Assume that $f^{-1}(\mathfrak{B})$ is a filter basis on X. (FB2) implies $f^{-1}(\mathfrak{B})$ is non-empty and empty subset of X does not belong to $f^{-1}(\mathfrak{B})$. For each $A \in \mathfrak{B}$, $f^{-1}(A) \in f^{-1}(\mathfrak{B})$.

So $f^{-1}(A) \neq \emptyset$.

Conversely, assume that $f^{-1}(A) \neq \emptyset$ for every $A \in \mathfrak{B}$. Since $\mathfrak{B}$ is a filter basis, $\mathfrak{B} \neq \emptyset$ and empty subset of Y does not belong to $\mathfrak{B}$. If $A \in \mathfrak{B}$, then $A \neq \emptyset$.

Moreover, $f^{-1}(A) \neq \emptyset$ for every $A \in \mathfrak{B}$.Therefore the empty subset of X does not belong to $f^{-1}(\mathfrak{B})$. Take $Z_1, Z_2 \in f^{-1}(\mathfrak{B})$.Then there exist $A_1, A_2 \in \mathfrak{B}$ such that $Z_1 = f^{-1}(A_1)$ and $Z_2 = f^{-1}(A_2)$. If $A_1, A_2 \in \mathfrak{B}$, then there exists $A_3 \in \mathfrak{B}$ such that $A_3 \subset A_1 \cap A_2$. It follows that $f^{-1}(A_3) \subset f^{-1}(A_1 \cap A_2) = f^{-1}(A_1) \cap f^{-1}(A_2) = Z_1 \cap Z_2$.Therefore $f^{-1}(\mathfrak{B})$is a filter basis on X, if $f^{-1}(A) \neq \emptyset$ for every $A \in \mathfrak{B}$.

6.4 Definition

Let A be a non-empty subset of a set X and $\mathcal{F}$ a filter on X. Then the **trace** of $\mathcal{F}$on A is defined and denoted by $\mathcal{F}_A = \{A \cap B: B \in \mathcal{F}\}$.

6.5 Proposition

If $\mathfrak{B}$ is a filter basis on X, then the trace $\mathfrak{B}_A = \{A \cap B: B \in \mathfrak{B}\}$ is a filter basis on A if and only if$A \cap B \neq \emptyset$ for every $B \in \mathfrak{B}$.

Proof

Let $f: A \rightarrow X$ be the canonical injection of A into X defined by $f(x) = x$.

Let $B \in \mathfrak{B}$.

$f^{-1}(B) = \{x \in A: f(x) \in B\} = \{x \in A: x \in B\} = A \cap B$.

$f^{-1}(\mathfrak{B}) = \{f^{-1}(B): B \in \mathfrak{B}\} = \{A \cap B: B \in \mathfrak{B}\} = \mathfrak{B}_A$.

Proposition 6.3 implies $f^{-1}(\mathfrak{B})$ is a filter basis if and only if $f^{-1}(B) \neq \emptyset$ for every $B \in \mathfrak{B}$. Therefore $\mathfrak{B}_A$ is a filter basis if and only if $A \cap B \neq \emptyset$ for every $B \in \mathfrak{B}$.

7. Convergence of Cauchy Filter

7.1 Definition

Let X be a topological space and $\mathfrak{B}$ a filter basis on X. A point x of X is said to **adhere** to $\mathfrak{B}$ if x adheres to every set A in $\mathfrak{B}$.

7.2 Definition

Let E be a topological vector space and $A \subset E$. A filter $\mathcal{F}$ on A is said to be a **Cauchy filter** if for every neighborhood of zero V, there exists a set $X \in \mathcal{F}$ such that $X - X \subset V$.

7.3 Proposition

Suppose that $\mathcal{F}$ is a filter on a set A of a topological vector space E and that $\mathcal{F}$ converges to a point $x \in E$. Then $\mathcal{F}$ is a Cauchy filter on A.

Proof

Assume that $\mathcal{F}$ on A converges to $x \in E$. Let V be neighborhood of zero in E. Then there exists a balanced neighborhood U of zero such that $U + U \subset V$. Since $\mathcal{F}$ converges to x, $\mathcal{B}(x) \subset \mathcal{F}$. Thus $x + U \in \mathcal{B}(x) \subset \mathcal{F}$. Then there exists $X \in \mathcal{F}$ such that $X \subset x + U$.

Let $z \in X - X$. Then there exists $y, w \in X$ such that $z = y - w$. Since $y, w \in X$ and $X \subset x + U$, $y - x$ and $w - x \in U$. Since U is balanced, $w - x \in U$ implies $x - w \in U$.

Thus $(y - x) + (x - w) \in U + U \subset V$. So $z = y - w \in V$, for every $z \in X - X$. Hence $X - X \subset V$. Therefore $\mathcal{F}$ is a Cauchy filter.

7.4 Proposition

If the point x adheres to the Cauchy filter $\mathcal{F}$ on a set A of topological vector space E, then $\mathcal{F}$ converges to x.

Proof

Let $\mathcal{F}$ be a Cauchy filter on A and x adheres to $\mathcal{F}$.

Take W $\in \mathcal{B}$(x), where$\mathcal{B}$(x) is the set of neighborhood of x.

Hence there exists a neighborhood V of zero such that x +V $\subset$ W.

Therefore there exists a neighborhood U of zero such that U+U $\subset$V. Since $\mathcal{F}$is Cauchy filter, there exists X $\in \mathcal{F}$ such that X–X $\subset$U. x$\in \overline{X}$since x adheres to X and x + U is a neighborhood of x and hence

(x+ U) $\cap$ X$\neq \emptyset$.

Take y $\in$ (x + U) $\cap$X .Then y $\in$ x + U and y $\in$ X.

Let z $\in$ X. Then z –y $\in$ X–X $\subset$U.So z $\in$ y+ U $\subset$ x + U +U$\subset$ x + V$\subset$ W. Hence X $\subset$ W.Since X $\in$ $\mathcal{F}$ and X $\subset$W, W$\in \mathcal{F}$ and $\mathcal{B}$(x) $\subset$ $\mathcal{F}$.

Therefore $\mathcal{F}$ converges to x.

Acknowledgements

I am very grateful to the Myanmar Academy of Arts and Science for allowing the presentation of this research paper. I would like to appreciate Dr Win Naing, Rector, Dr Sein Sein Aung and DrThetNaingOo, Pro-Rectors of Monywa University, for their suggestions. I am also very thankful to Dr Zaw Myint, Professor and Head, Department of Mathematics, Monywa University, for his valuable useful comments.

References

1. Clark, P., "Convergence", Department of Mathematics, University of Georgia, October 18, 2016.
2. Galvin, D., "Ultrafilters, with applications to analysis, social choice and combinatorics", Department of Mathematics, University of Notre Dame, September 3, 2009.
3. Garcia, M., "Filters and Ultrafilters in Real Analysis", Department of Mathematics, California Polytechnic State University, Related articles, December 22, 2012.
4. Horvath, J., "Topological Vector Spaces and Distributions", Volume 1, Addison Wesley Publishing Company, London, 1966.
5. Kruchman, A., "Note on Ultrafilters", Prepared for Berkeley Math Toolbox Seminar, November 7, 2012.
6. Thein Myint, Dr., "Lecture Notes on Functional Analysis", Department of Mathematics, University of Mandalay, 2011.
7. T. Neve., "Theorems on Ultrafilters", Leiden University Bachelor Project Related Articles, August 25, 2013.

COMPARISON OF SIMPLE ACCELERATION METHOD AND CONWAY'S METHOD

Li Li Than*

Abstract

Congruence can be used to determine on which day of the week a given date falls. We discuss the calendar formula to calculate the day of the week of December 25, 2017. Also, we express easier formula to calculate. And then, we describe Conway's Doomsday Algorithm. The Doomsday Algorithm is an Algorithm for calculating the day of the week for any given calendar date. The algorithm is based on first computing doomsday which is the day of the week of the last day of February, or of January. We present an acceleration method for calculating the dooms year term of the Doomsday algorithm. Finally, we show that simple acceleration method is similar in form to Conway's lookup table acceleration method.

1. Definitionsm

A year is the amount of time it takes the Earth to make one complete orbit around the Sun.

A **day** is the amount of time it takes the Earth to make a complete rotation about the axis through its north and south poles.

A year is approximately 365.2422 days long. In 46 B.C., Julius Caesar (and its scientific advisors) compensated for this by creating the **Julian calendar**, containing a **leap year** every 4 years; that is, every fourth year has an extra day, namely, February 29, and so it contains 366 days. A **common year** is a year that is not a leap year.

This would be fine if the year were exactly 365.25 days long, but it has the fact of making the year 365.25 − 364.2422 = .0078 (about 11 minutes 14 seconds) days too long. After 128 years, a full day was added to the calendar, that is, the Julian calendar over counted the number of days.

Let us now seek a calendar formula. For easier calculation, we choose 0000 as our reference year, even though there was no year! Assign a number to each day of the week, according to the following scheme:

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
0	1	2	3	4	5	6

*. Dr, Lecturer, Department of Mathematics, Myingyan Degree Collage

In particular, March 1, 0000, has some number a , where $0 \le a \le 6$. In the next year, 0001, March 1 has number a +1(mod 7), for 365 days have elapsed from March 1, 0000, to March 1, 0001, and

$$365 = 52\times7 +1\equiv1 \text{ mod}7.$$

Similarly, March 1, 0002, has number a + 2, and March 1, 0003, has number a +3. However, March 1, 0004, has number a +5 , for February 29, 0004, fell between March 1, 0003, and March 1, 0004, and so $366 \equiv 2\text{mod}7$ days had elapsed since the previous March 1. We see, therefore, that every common year adds 1 to the previous number for March 1, while each leap year adds 2. Thus, if March 1, 0000, has number a, then the number a′ of March 1, year y, is

$$a' \equiv a + y + L \text{mod}7,$$

[For 0000, $365 \equiv a \text{mod } 7$,
for 0001, $2 \times 365 \equiv a + 1 \text{mod } 7$,
for 0002, $3 \times 365 \equiv a + 2 \text{mod } 7$,
for 0003, $4 \times 365 \equiv a + 3 \text{mod } 7$,
for 0004, $4 \times 365 \equiv a + 4 + 1 \text{mod } 7$]

where L is the number of leap years from year 0000 to year y. To compute L, count all those years divisible by 4, then throw away all the century years, and then put back those back century years that are leap years. Thus,

$$L \equiv \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor,$$

where $\lfloor x \rfloor$ denotes the greatest integer in x.

For 1 year, L = 0,
for 2 years, L = 0,
for 3 years, L = 0,
for 4 years, L = 1,
and so on for 100 years , L = 24 = 100/ 4 -100/100,
for 400 years, L = 97 = 400/4 -400/100+400/400.

Therefore, we have

$$\begin{aligned} a' &\equiv a + y + L \\ &\equiv a + y + \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor \text{ mod } 7. \end{aligned}$$

We can actually find a′ by looking at a calendar. Since March 2, 1994, fell on a Tuesday,

$$\begin{aligned} 2 &\equiv a + 1994 + \lfloor 1994/4 \rfloor - \lfloor 1994/100 \rfloor + \lfloor 1994/400 \rfloor \\ &\equiv a + 1994 + 498 - 19 + 4 \bmod 7, \end{aligned}$$

and so

$$a \equiv -2475 \equiv -4 \equiv 3 \bmod 7$$

(that is, March 1, year 0000, fell on Wednesday). One can now determine the day of the week on which March 1 will fall in any year $y > 0$, namely, the day corresponding to

$$3 + y + \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor \bmod 7.$$

There is a reason we have been discussing March 1.

Let us now analyze February 28. For example, suppose that February 28, 1600, has number b. As 1600 is a leap year, February 29, 1600, occurs between February 28,1600, and February 28, 1601; hence 366 days have elapsed between these two February 28's, so that February 28, 1601, has number b+2. February 28, 1602, has number b+3, February 28, 1603, has number b+4, February 28, 1604, has number b+5, but February 28, 1605, has number b+7 (for there was a February 29 in 1604).

Let us compare the pattern of behavior of February 28, 1600, namely,

b, b+2, b+3, b+4, b+5, b+7, ..., with that of some date in 1599. If May 26, 1599, has the number c, then May 26, 1600, has the number c+2, for February 29, 1600, comes between these two May 26's, and so there are $366 \equiv 2 \bmod 7$ intervening days. The numbers of the next few May 26's, beginning with May 26, 1601, are c, c + 2, c + 3, c + 4, c + 5, c +7. We see that the pattern of the days for February 28, starting in 1600, is exactly the same as the pattern of the days for May 26, starting in 1599; indeed, the same is true for any date in January or February. Thus, the pattern of the days for any date in January or February of a year y is the same as the pattern for a date occurring in the preceding year y −1: a year preceding a leap year adds 2 to the number for such a date, whereas all other years add 1. Therefore,

February 28, 1600, has number b,

February 28, 1601, has number b + 2, (since 1600 is a leap year)

February 28, 1602, has number b + 3,

February 28, 1603, has number b + 4,

February 28, 1604, has number b + 5,

February 28, 1605, has number b+ 7, (since 1604 is a leap year)

So, it has the pattern: b, b+2, b+3, b+4, b+5, b+7,...,

May 26, 1599, has the number c,

May 26, 1600, has the number c + 2, (for February 29, 1600)

May 26, 1601, has the number c + 3,

May 26, 1602, has the number c + 4,

May 26, 1603, has the number c + 5,

May 26, 1604, has the number c + 7,

so it has the pattern:

$$c, c + 2, c + 3, c + 4, c + 5, c + 7.$$

Now we find the day corresponding to a date other than March 1. Since March 1,0000, has number 3, April 1, 0000, has number 6, for March has 31 days and $3+31 \equiv 6 \bmod 7$. Since April has 30 days, May 1, 0000, has number $6+30 \equiv 1 \bmod 7$. Thus, the following table gives the number of the first day of each month in year 0000:

March 1,0000, has number	3
April 1	6
May 1	1
June 1	4
July 1	6
August 1	2
September 1	5
October 1	0
November 1	3
December 1	5
January 1	1
February 1	4

We are pretending that March is month 1, April month 2, etc. Let us denote these numbers by $1+j(m)$, where $j(m)$, for $m=1,2,\ldots,12$, is defined by

$$j(m) : 2,5,0,3,5,1,4,6,2,4,0,3.$$

$$\begin{array}{lccccccc}
 & \text{day} & & j(m) & & \text{year} & & \\
\text{For March 1, 0000 has number} & 1 & + & 2 & + & 0 & = 3 & \\
\text{For March 1, 0001 has number} & 1 & + & 2 & + & 1 & = 4 & \\
\text{For March 1, 0002 has number} & 1 & + & 2 & + & 2 & = 5 & \\
\text{For March 1, 0003 has number} & 1 & + & 2 & + & 3 & = 6 & \\
\text{For March 1, 0004 has number} & 1 & + & 2 & + & 4 & +1 = 8 \equiv 1 \mod 7 & \\
 & 1 & + & 2 & + & 4+\lfloor 4/4 \rfloor - \lfloor 4/100 \rfloor + \lfloor 4/400 \rfloor \equiv 1 \mod 7 & &
\end{array}$$

$$\vdots$$

$$\text{For month m, day d, and year y,} \quad d + j(m) + g(y) \mod 7$$

$$\text{where } g(y) = y + \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor.$$

2. Calendar Formula and Its Application

Proposition 2.1. (Calendar Formula). The date with month m, day d, year y has number

$$d + j(m) + g(y) \mod 7 ,$$

where

$$j(m) = 2,5,0,3,5,1,4,6,2,4,0,3,$$

(March corresponds to m =1, April to m = 2, , February to m =12) and

$$g(y) = y + \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor,$$

provided that dates in January and February are treated as having occurred in the previous year.

Proof. The number mod 7 corresponding to month m, day 1, year y, is $1+ j(m) +g(y)$. It follows that $2+ j(m) +g(y)$ corresponds to moth m, day 2, year y, and, more generally, that $d + j(m) +g(y)$ corresponds to month m, day d, year y.

Example 2.2. Let us use the calendar formula to find the day of the week on which December 25, 2017, fell. Here $m= 4, d = 25$,and $y = 2017$. Substituting in the formula, we obtain the number

$$25+ 4+ 2017+\lfloor 2017/4 \rfloor - \lfloor 2017/100 \rfloor + \lfloor 2017/400 \rfloor = 2535 \equiv 1 \bmod 7,$$

therefore, December 25, 2017, fell on a Monday.

Most of us need paper and pencil (or calculator) to use the calendar formula in the proposition. Now we use some ways to calculation.

One mnemonic for j(m) is given by

$$j(m) = \lfloor 2.6m - 0.2 \rfloor, \text{ where } 1 \le m \le 12.$$

In above example, we also obtain the number

$$1+\lfloor (2.6)10 - 0\cdot 2 \rfloor + 2017 + \lfloor 2017/4 \rfloor - \lfloor 2017/100 \rfloor + \lfloor 2017/400 \rfloor = 2535 \equiv 1 \bmod 7$$

where m =10.

Another mnemonic for j(m) is in the sentence:

My	Uncle	Charles	has	eaten	a	cold	supper;	he	eats	nothing	hot.
2	5	$(7 \equiv 0)$	3	5	1	4	6	2	4	$(7 \equiv 0)$	3

2.3 Corollary. The date with month m, day d, year $y = 100C + N$, where $0 \le N \le 99$, has the number

$$d + j(m) + N + \lfloor N/4 \rfloor + \lfloor C/4 \rfloor - 2C \bmod 7,$$

provided that dates in January and February are treated as having occurred in the previous year.

Proof. If we write a year $y = 100C + N$, where $0 \le N \le 99$, then

$$y = 100C + N \equiv 2C + N \bmod 7,$$

$$\lfloor y/4 \rfloor = 25C + \lfloor N/4 \rfloor \equiv 4C + \lfloor N/4 \rfloor \bmod 7,$$

$$\lfloor y/100 \rfloor = C, \text{ and } \lfloor y/400 \rfloor = \lfloor C/4 \rfloor.$$

Therefore,

$$\begin{aligned} y + \lfloor y/4 \rfloor - \lfloor y/100 \rfloor + \lfloor y/400 \rfloor &\equiv 2C + N + 4C + \lfloor N/4 \rfloor - C + \lfloor C/4 \rfloor \bmod 7 \\ &\equiv N + \lfloor N/4 \rfloor + \lfloor C/4 \rfloor - 2C \bmod 7. \end{aligned}$$

This formula is simpler than the first one. For example, the number corresponding to December 25, 2017, is now obtained as

$$25 + 4 + 17 + \lfloor 17/4 \rfloor + \lfloor 20/4 \rfloor - 2(20) = 15 \equiv 1 \bmod 7.$$

Now I find the day of my birthday.

2.4. Example My birthday date is June 27, 1973. On what day of the week was I born?

If A is the number of the day, then

$$A \equiv 27 + 3 + 73 + \lfloor 73/4 \rfloor + \lfloor 19/4 \rfloor - 2(19) = 87 \equiv 3 \bmod 7.$$

I was born on a Wednesday.

3. Conway's Calendar Formula

John Horton Conway has found an even simpler calendar formula. In his system, he calls doomsday of a year that day of the week on which the last day of February occurs. For example, doomsday 1900, corresponding to February 28, 1900(1900 is not a leap year), is Wednesday = 3, while

doomsday 2000, corresponding to February 29, 2000, is Tuesday = 2, as we know from Corollary 2.3.

Knowing the doomsday of a century year 100C, we can find the doomsday of any other year y = 100C + N in that century, as follows. Since 100C is a century year, the number of leap years from 100C to y does not involve the Gregorian alteration. Thus, if D is doomsday 100C (of course, $0 \leq D \leq 6$), then doomsday 100C + N is congruent to

$$D + N + \lfloor N/4 \rfloor \bmod 7.$$

For example, since doomsday 1900 is Wednesday = 3, we see that doomsday 1994 is Monday =1, for

$$3 + 94 + 23 = 120 \equiv 1 \bmod 7.$$

3.1 Proposition. (Conway's Formula). Let D be doomsday 100C, and let $0 \leq N \leq 99$. If $N = 12q + r$, where $0 \leq r < 12$, then the formula for doomsday 100C + N is

$$D + q + r + \lfloor r/4 \rfloor \bmod 7.$$

Proof.

$$\begin{aligned} \text{Doomsday}(100C + N) &\equiv D + N + \lfloor N/4 \rfloor \\ &\equiv D + 12q + r + \lfloor (12q + r)/4 \rfloor \\ &\equiv D + 15q + r + \lfloor r/4 \rfloor \\ &\equiv D + q + r + \lfloor r/4 \rfloor \bmod 7 \end{aligned}$$

For example, $94 = 12 \times 7 + 10$, so that doomsday 1994 is $3 + 7 + 10 + 2 \equiv 1 \bmod 7$; that is, doomsday 1994 is Monday, as we saw above.

We know doomsday of a particular year, we can use various tricks (e.g., my Uncle Charles) to pass from doomsday to any other day in the year. Conway observes that some other dates falling on the same day of the week as the doomsday are

April 4, June 6, August 8, October 10, December 12,

May 9, July 11, September 5, and November 7;

it is easier to remember these dates using the notation

4/4, 6/6, 8/8, 10/10, 12/12, and 5/9, 7/11, 9/5, and 11/7,

where m/d denotes month/day (we now return to the usual counting having January as the first month :1 = January). Since doomsday corresponds to the last day of February, we are now within a few weeks of any date in the calendar, and we can easily interpolate to find the desired day.

4. The Doomsday Algorithmas a poem

John Conway introduced the Doomsday Algorithm with the following rhyme:

(1) The last of February or of January will do

(2) (Except that in Leap Years it's January 32).

(3) Then for even months use the month's own day,

(4) And for odd ones add 4, or take it away.

(5) Now to work out your doomsday the orthodox way

(6) Three things you should add to the century day

(7) Dozens, remainder, and fours in the latter,

(8) (If you alter by sevens of course it won't matter)

(9) In Julian times, lackaday, lackaday

(10) Zero was Sunday, centuries fell back a day

(11) But Gregorian 4 hundreds are always Tues.

(12) And now centuries extra take us back twos.

(13) According to length or simply remember,

(14) you only subtract for September, or November.

4.1 The Doomsday

In other words here is a simple trick that we can use to determine the day of the week for any date of the current year. The day of the week on which the last day of February falls is called the **doomsday**. For non-leap years (or common years), this date is February 28; for leap years, it is February 29. Since 2017 is a common year, the current doomsday is the day of the week on which February 28, 2017 occurred; a Tuesday. Everything else that we need follows from one simple lemma:

4.2 Lemma

Adding or subtracting any integer multiple of 7 to any date leaves the day of the week unchanged. For example, February 7, 14, 21, and 28 all fall on the same day of the week. Likewise, adding x days is equivalent to adding x – 7 days, which is equivalent to subtracting 7– x days. For example, with x = 6, the day of the week that falls 6 days after Monday, is the same as the one that is 7– 6 = 1 day before.

4.3 The rule of January

We now use the Lemma 4.2 to identify at least one date in every other month that falls on the same day of the week as the doomsday. We begin with the **rule of January**. In a common year (like 2013) February has 28 days. By applying the above lemma, we subtract 4 7 = 28 days from February 28 to arrive at February 0, which must also fall on the doomsday. But February 0 is just another name for January 31 as both dates immediately precede February 1. Thus in a common year, January 31 is the doomsday. In leap years, the doomsday is February 29. Again subtracting 28, a multiple of 7, yields February 29 – 28 = February 1. Thus in a leap year, February 1 falls on the doomsday. With a touch of whimsey, this date is also called "January 32", as both dates immediately follow January 31. Thus in leap years January 32 is the doomsday. Alternatively, observe that

$$31 - 28 = 3, \text{ and } 32 - 28 = 4.$$

Thus January 3 always falls on the same day of the week as January 31; as January 4 does for February 1. Consequently, for common years (which

come in groups of 3), the doomsday is January 3; and for leap years (which come once every 4 years) the doomsday is January 4.

4.4 The rule of March

Since the doomsday immediately precedes March 1(for both leap years and non-leap years), we call the last day of February "March 0". Thus, for every year, March 0 is the doomsday. If we insist on using an actual date in March, Lemma 4.2 implies that March 7, 14, 21, and 28 any multiple of 7) are all doomsdays.

4.5 The rule of even months

The third line of Conway's rhyme expresses the **rule of even months**. Thus for April, the fourth month, the doomsday is on 4/4. For June it is 6/6; August, 8/8; October 10/10; and December, 12/12. The answer follows from Lemma 4.2 and an interesting pattern within the seemingly irregular distribution

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
31	28 29	31	30	31	30	31	31	30	31	30	31

Conway observed that when eight of the months are paired as shown (April with May, June with July, August with September, and October with November), then together each pair contains exactly 30 + 31 =61 days. Since adding 2 to 61 produces a multiple of 7, every doomsday in June occurs two dates after the corresponding doomsdays in April. Likewise every doomsday in August occurs two dates after a doomsday in June; etc. So all we need to do is identify one doomsday in April the remainder of even months will fall like dominoes. Using Lemma 4.2 with the rule of March we learn that March 35 is a doomsday.

To convert March 35, we carry (just like in ordinary arithmetic) 31 days (the number of days in March) from the date column, and advance the month. Thus,

$$\text{March } 35=\text{April}(35-31)=\text{April } 4.$$

Likewise doomsdays fall on June 4+2=6, August 6+2=8, October 8 + 2 = 10, and December 10 + 2 = 12, demonstrating the rule of even months.

4.6 The rule of odd months

The fourth line and footnote in Conway's rhyme describe the **rule of odd months**. Adding 4to the index of every odd month having 31 days; and subtracting 4 from the index of those that have only 30 days, yield the remaining doomsdays:

$$\begin{aligned}
&\text{Mar } 3+4 = \text{Mar } 7, \text{ or } 3/7,\\
&\text{May } 5+4 = \text{May } 9, \text{ or } 5/9,\\
&\text{Jul } 7+4 = \text{Jul } 11, \text{ or } 7/11,\\
&\text{Sep } 9-4 = \text{Sep } 5, \text{ or } 9/5,\\
&\text{Nov } 11-4 = \text{Nov } 7, \text{ or } 11/7,
\end{aligned}$$

Note that rule of odd months is consistent with the rule of March, and that for the remaining four months, 9is paired with 5 (5/9 and 9/5), while 7 is always paired with 11 (7/11 and 11/7). We can thus use the mnemonic "working 9 to 5 at the 7-Eleven," the latter being a national convenience store chain.

The rationale for the rule of odd months follows for each odd month: From the rule of March and Lemma 4.2, March 7, is a doomsday, Thus, adding 4 to the index of March (3) yields the doomsday March 7 =3/7.
Next, we add 63 (a multiple of 7) to March 7, obtaining March 70. Carrying the months of March and April in succession, yields

$$\text{March } 70 = \text{April}(70-31) = \text{May}(70-31-30) = \text{May } 9, \text{or} 5/9.$$

Advancing another 63 days, yields the doomsday May 72. Again we carry two months,

$$\text{May } 72 = \text{June}(72-31) = \text{July}(72-31-30) = \text{July } 11, \text{or } 7/11.$$

$$\text{Jul}67 = \text{Aug}(67-31) = \text{July}(67-31-31) = \text{Sep}5, \text{or } 9/5.$$

$$\text{Sep}68 = \text{Oct}(68-30) = \text{July}(68-30-31) = \text{Nov}7, \text{or } 11/7.$$

See Table 1 for a summary of the doomsdays obtained for each month.

5. Finding the doomsday in a future or past year

The second stanza of Conway's poem describes how to find the doomsday for an arbitrary year. The basic fact to remember is that common years, like 2011, 2013, and 2014, have exactly 365 days. It is easy to verify that $365 = 7 \times 52 + 1$. Thus if the following year is a common year, then the doomsday advances by one day of the week. So in 2014 the last day of February, February 28, 2014, will fall on a Friday, and all of the dates shown in Table 1 will be Fridays in the year 2014. Likewise, if the current year is a common year, then the doomsday of the previous year retreats by one day of the week. Thus, the doomsday for 2012 (February 29, 2012) was a Wednesday.

Leap years on the other hand have $366 = 7 \times 52 + 2$ days. Thus if the following year is a leap year, then its doomsday will advance by two days of the week. And if the current year is a leap year then the previous year's doomsday would be two days earlier in the week. Thus, the doomsday of 2011 was Wednesday – 2 = Monday. The following table (in which leap years appear in bold typeface) illustrates this.

Doomsdays				
Jan. $\begin{Bmatrix}3 \text{ or } 31\\4 \text{ or } 32\end{Bmatrix}$	Rule of January		Jul. 11	Rule of odd months (7+4)
Feb. $\begin{Bmatrix}28\\29\end{Bmatrix}$	Basic definition		Aug. 8	Rule of even months (8/8)
Mar. 7	Rule of odd months (3+4)		Sept. 5	Rule of odd months (9−4)
Apr. 4	Rule of even months (4/4)		Oct. 10	Rule of even months (10/10)
May. 9	Rule of odd months (5+4)		Nov. 7	Rule of odd months (11−4)
Jun. 6	Rule of even months (6/6)		Dec. 12	Rule of even months (12/12)

Table 1: A summary of the doomsday rules applied to each month of the year. For those dates appearing in curled braces, the upper value should be used in a non-leap year, and the lower, in a leap year.

Year		**1988**	1989	1990	1991	**1992**	1993.	1994	1995	**1996**	1997	1998	1999
Doomsday		Mon.	Tue.	Wed.	Thu.	Sat.	Sun.	Mon.	Tue.	Thu.	Fri.	Sat.	Sun.
Year		**2000**	2001	2002	2003	**2004**	2005	2006	2007	**2008**	2009	2010	2011
Doomsday		Tue.	Wed.	Thu.	Fri.	Sun.	Mon.	Tue.	Wed.	Fri.	Sat.	Sun.	Mon.
Year		**2012**	2013	2014	2015	**2016**	2017	2018	2019	**2020**	2021	2022	2023
Doomsday		Wed.	Thu.	Fri.	Sat.	Mon.	Tue.	Wed.	Thu.	Sat.	Sun.	Mon.	Tue.

Table 2: The day of the week on which the doomsdays listed in Table 1 fall on. Leap years are identified in bold font

6. The twelve-year rule

Also note any 12-year jump forward (up to the 99th year in a century) advances the doomsday by one day of the week for both leap or non-leap years. Actually, we don't need the table to figure this out. Every such 12year period contains exactly 3 leap years, and therefore exactly 12 – 3=9 non-leap years. So moving forward by twelve years advances the doomsday by 3×2+9×1=15 days. Subtracting two sets of 7(remember adding or subtracting 7 does not change the weekday) yields 15−14 =1. So the day advances by 1, and thus the doomsday in 2026will fall on a Saturday. Going backwards by 12 results in a 1 day retreat, so the doomsday in2014−12=2002 was Thursday. We'll call this the twelve-year rule.

7. Computing the doomsday for an arbitrary year

To simplify computing the doomsday for years in different centuries, Conway's algorithm uses the last year of each century as a reference. It is not difficult to verify that the doomsdays for these years obey the following pattern, (see lines 11 and 12 in the poem):

GREGORIAN CENTURIES BY DOOMSDAY						
SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1700		1600	1500			
2100		2000	1900		1800	
2500		2400	2300		2200	

Note that every Gregorian century mark divisible by 400 is a leap year, and has a doomsday of Tuesday. Furthermore, the doomsday retreats by two weekdays with every advancing (non-leap year) century. The ultimate short cut is expressed in the second stanza (lines 5–8). For July 4, 1776. Start with the century mark 1600; the doomsday is Tuesday. Moving forward to 1700, the doomsday falls back two days to Sunday. Now find the largest multiple of 12 that is less than or equal to 76 (that is 1776 − 700). Clearly 76 = 12 × 6 + 4. So the doomsday advances 6 days, for the quotient, plus 4 days for the remainder, plus 1 more day because 1776 is in fact a leap year. Thus the doomsday of 1776 is Sunday plus 11 days, which by the lemma equals Sunday minus three days, or Thursday. Since July 4, is always a doomsday, July 4, 1776 was a Thursday.

Sometimes we may see the notation ⌊76/12⌋=6, which means that the greatest integer contained in the quotient 76/12 =6.333...is 6. The function ⌊x⌋is called the **floor of x**. Likewise, we frequently represent the remainderby the mod, or **modulus**. Explicitly76 mod 12 =4. Consequently, the entire doomsday calculation for July 4, 1776 can be written as

Sunday+⌊76/12⌋+76mod12+⌊(76mod12)/4⌋=Sunday+6+4+1=Sunday +11=Thursday.

Finally, for the Julian calendar (which was still used in English speaking countries and colonies up to 1750), the doomsdays retreated by one day every century.

JULIAN CENTURIES BY DOOMSDAY						
SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
0						
700	600	500	400	300	200	100
1400	1300	1200	1100	1000	900	800
				1700	1600	1500

8. Methods for Accelerating Conway's Doomsday Algorithm

We propose a modification of a key component in the Doomsday Algorithm for calculating the day of the week of any given date. In particular, we propose to replace the calculation of the required expression:

$$\left\lfloor \frac{x}{12} \right\rfloor + x \bmod 12 + \left\lfloor \frac{x \bmod 12}{4} \right\rfloor$$

with

$$2y + 10(y \bmod 2) + z + \left\lfloor \frac{2(y \bmod 2) + z}{4} \right\rfloor$$

where x is an input 2-digit year;

y is the tens digit of x;

z is the ones digit of x;

We argue the fact that our modification operates on individual base-10 digits makes the algorithm easier to calculate mentally.

The Doomsday algorithm's input is a calendar date of the form MM/DD/YYYY where MM is the month, DD is the day, and YYYY is the year. YYYY can further be broken down to its constituent century CC and year within the century YY. The output of the algorithm is a number between 0 to 6 that corresponds to each of the 7 days for the week.

The key equation of the Doomsday algorithm can be described a sum (modulo 7) of three terms:

day_of_the_week = (doomscentury + doomsyear + doomsmonth) mod 7

where:

doomscentury(CC) is a function of the input date's century

doomsyear(YY) is a function of the input date's 2-digit year within a century

doomsmonth(MM,DD) is a function of the input date's calendar month and day.

The doomsyear formula provided by Conway is:

$$\left\lfloor \frac{x}{12} \right\rfloor + x \bmod 12 + \left\lfloor \frac{x \bmod 12}{4} \right\rfloor$$

where x is the input date's 2-digit year within a century. The addition is always modulo 7, so the resulting sum is a number between 0 and 6, inclusive.

8.1 Simple Method

Let us break down the two-digit year x into its constituent digits y and z, where y is the tens digit, and z is ones digit. That is,

$$y = \left\lfloor \frac{x}{10} \right\rfloor$$

$$z = x \bmod 10$$

For example, if x = 74, then y= 7 and z = 4. If x = 88, then y= 8 and z = 8.

Having defined y and z in terms of x, we propose the replacement doomsyear function as

$$\text{doomsyear}(y,z) = 2y + 10(y \bmod 2) + z + \text{leaps}$$

where (y mod 2) is really just a decision function to tell whether y is odd or even.

$$(y \bmod 2) = 1 \text{ if } y \text{ is odd}$$

$$(y \bmod 2) = 0 \text{ if } y \text{ is even}$$

We define an extra variable called leaps as the number of leap years between the start of the y decade and the z year. If the start of a decade is a leap year, we don't count it. But if the year z is a leap year, we do include it. For example, if x = 88, the decade starts at 80 and we have leaps = 2 because 84 and 88 are leap years. If x = 74, the decade starts at 70 and we have leaps = 1 because 72 is a leap year. In general, the variable leaps can only have three values: 0, or 1, or 2. There can't be more than 2 leap years after the start of a decade. Remember, we never include the start of the decade in our leap count.

The explicit formula for leaps is

$$\text{leaps} = \left\lfloor \frac{2(y \bmod 2) + z}{10} \right\rfloor$$

8.2 Examples.

Let's calculate the doomsyear term for these years:

1) 1974: y = 7, z = 4
   doomsyear = 2*7 + 10*1 + 4 + leaps = 14 + 10 + 4 + 1 = 29 = **1**(mod 7)
   leaps = 1 because 1972 is a leap year
2) 2040: y = 4, z= 0
   doomsyear = 2*4 + 10*0 + 0 + leaps = 8 + 0 + 0 + 0 = **1**(mod 7)
   leaps = 0
3) 2010: y = 1, z = 0
   doomsyear = 2*1 + 10*1 + 0 + leaps = 2 + 10 + 0 + 0 = 12 = **5**(mod 7)
   leaps = 0
4) 1988: y = 8, z= 8
   doomsyear = 2*8 + 10*0 + 8 + leaps = 16 + 0 + 8 + 2 = 26 = **5**(mod 7)
   leaps = 2 because 1984 and 1988 are leap years
5) 2007: y = 0, z= 7
   doomsyear = 2*0 + 10*0 + 7 + leaps = 0 + 0 + 7 + 1 = 8 = **1**(mod 7)
   leaps = 1 because 2004 is a leap year
6) 1998: y = 9, z=8
   doomsyear = 2*9 + 10*1 + 8 + leaps = 18 + 10 + 8 + 2 = 38 = **3** (mod 7)
   leaps = 2 because 1992 and 1996 are leap years.

Let's define the decade anchor to be the 2y + 10(y mod 2) subexpression of our doomsyear term. This subexpression only depends on the y decade of the input year.

$$\text{decade\_anchor}(y) = (\, 2\, y + 10\, (y \bmod 2)\,) \bmod 7$$

Here is the table to memorize:

y	decade	2y + 10(y mod 2)	decade anchor
0	00's	0	0
1	10's	12	5
2	20's	4	4
3	30's	16	2
4	40's	8	1
5	50's	20	6
6	60's	12	5
7	70's	24	3
8	80's	16	2
9	90's	28	0

Table : Decade anchor lookup

If we memorize this simple table of 10 numbers, we can avoid calculating the decade anchor 2y + 10 (y mod 2) component of the doomsyear formula.

Thedoomsyeaformula is thus:

$$doomsyear(y,z) = decade\_anchor(y) + z + leaps$$

Let's look at a table of possible values for the leap term depending on the digit year z:

digit year z	leaps	leaps if y is even	leaps if y is odd
0	0	0	0
1	0	0	0
2	0 or 1	0	1
3	0 or 1	0	1
4	1	1	1
5	1	1	1
6	1 or 2	1	2
7	1 or 2	1	2
8	2	2	2
9	2	2	2

Table : Possible Values for leap

year	Conway's doomsyear	2y + 10 (y mod 2) + z + leaps	year	Conway's doomsyear	2y + 10 (y mod 2) + z + leaps
0	0	0	56	0	0
1	1	1	57	1	1
2	2	2	58	2	2
3	3	3	59	3	3
4	5	5	60	5	5
5	6	6	61	6	6
6	0	0	62	0	0
7	1	1	63	1	1
8	3	3	64	3	3
9	4	4	65	4	4
10	5	5	66	5	5
11	6	6	67	6	6
12	1	1	68	1	1
13	2	2	69	2	2
14	3	3	70	3	3
15	4	4	71	4	4
16	6	6	72	6	6
17	0	0	73	0	0
18	1	1	74	1	1
19	2	2	75	2	2
20	4	4	76	4	4
21	5	5	77	5	5
22	6	6	78	6	6
23	0	0	79	0	0
24	2	2	80	2	2
25	3	3	81	3	3
26	4	4	82	4	4
27	5	5	83	5	5
28	0	0	84	0	0
29	1	1	85	1	1
30	2	2	86	2	2
31	3	3	87	3	3
32	5	5	88	5	5
33	6	6	89	6	6
34	0	0	90	0	0
35	1	1	91	1	1
36	3	3	92	3	3
37	4	4	93	4	4
38	5	5	94	5	5
39	6	6	95	6	6
40	1	1	96	1	1
41	2	2	97	2	2
42	3	3	98	3	3
43	4	4	99	4	4
44	6	6			
45	0	0			
46	1	1			
47	2	2			
48	4	4			
49	5	5			
50	6	6			
51	0	0			
52	2	2			
53	3	3			
54	4	4			
55	5	5			

Table 3: Doomsyear Values from 00 to 99

8.3 Conway's Look-up Table Acceleration Method

John Horton Conway devised an acceleration method to speed-up the calculation of the doomsyear term. In practice, Conway's method is probably the fastest acceleration method for such, but it involves memorizing 18 numbers and some non-intuitive rules. Now, we will describe Conway's look-up table method. And then, we will compare and contrast our method with Conway's acceleration method.

Conway's lookup table method requires memorizing the years of the century where the doomsyear value is zero. Let us call these numbers as zero-anchor years. These are:

0	6	11.5	17	23	28	34	39.5	45
51	56	62	67.5	73	79	84	90	95.5

There's actually the added complication of the half-numbers: 11.5, 39.5, 67.5 and 95.5. These half-numbers mean that the preceding year has doomsyear value 6 and the succeeding year has doomsyear value 1. For example, doomsyear (11) = 6, and doomsyear(12) = 1; doomsyear (67) = 6, and doomsyear (68) = 1. These half-numbers occur because of the increment-by-2 property of doomsyear values during leap years. In effect, a doomsyear of value 0 got skipped in the half-number locations.

Here are the steps of Conway's acceleration method:

1) Select the nearest zero-anchor year less than your input year.

2) Let z_0 be the difference between your input year and the selected zero-anchor. Ignore fractional values of half-numbers in zero-anchor years. That is, treat 11.5, 39.5, 67.5, and 95.5 as 11, 39, 67, and 95 respectively in calculating the difference

3) Count the number of leap years between the zero-anchor and your input year. If the zero-anchor year is a leap year, do not include it. On the other hand, if your input year is a leap year, include it in the leap count. Let us denote this count of leap years as $leap_0$

4) Add z_0 and leap_0 to get the doomsyear value. If the selected zero-anchor is a half- number, subtract 1 from the sum. We denote this as the anchor adjustment term needed for half-number zero-anchors.

To sum it all up, Conway's acceleration method can be described by the equation:

$$\text{doomsyear} = \text{anchor\_adjustment} + z_0 + \text{leap}_0$$

Note that the anchor_adjustment term is almost always zero except for half-number years where it has a value of –1.

8.4 Examples

1) 1974:
   zero-anchor is 1973
   $z_0 = 1974 - 1973 = 1$
   $\text{leap}_0 = 0$
   doomsyear = 1 + 0 = **1**
2) 2040:
   zero-anchor is 2039.5
   $z_0 = 2040 - 2039 = 1$
   leap_0 =1 because 2040 is a leap year
   doomsyear = – 1 + 1 + 1 = **1**
3) 2010:
   zero-anchor is 2006
   $z_0 = 2010 - 2006 = 4$
   leap_0 = 1 because 2008 is a leap year
   doomsyear = 4 + 1 = **5**
4) 1988:
   zero-anchor is 1984
   $z_0 = 1988 - 1984 = 4$
   leap_0 = 1 because 1988 is a leap year. Remember, we don't count the zero-anchor
   doomsyear = 4 + 1 = **5**

5) 2007:
zero-anchor is 2006
$z_0 = 2007 - 2006 = 1$
$leap_0 = 0$
doomsyear = 1 + 0 = **1**

6) 1998:
zero-anchor is 1995.5
$z_0 = 1998 - 1995 = 3$
$leap_0 = 1$ because 1996 is a leap year
doomsyear = – 1 + 3 + 1 = **3**

7) 1914:
zero-anchor is 1911.5
$z_0 = 1914 - 1911 = 3$
$leap_0 = 1$ because 1912 is a leap year
doomsyear = – 1 + 3 + 1 = **3**

8) 1972:
zero-anchor is 1967.5
$z_0 = 1972 - 1967 = 5$
$leap_0 = 2$ because 1968 and 1972 are leap years
doomsyear = – 1 + 5 + 2 = **6**

9. A Comparison of Simple Acceleration Method with Conway's

Simple method is amenable to lookup table acceleration. In fact, we claim that after this lookup table acceleration, simple method is very similar in form to Conway's acceleration method. Let us compare and contrast the doomsyear equation for simple method and Conway's method. These are:

$$\text{doomsyear}(y,z) = \text{decade\_anchor}(y) + z + \text{leaps}$$

and

$$\text{doomsyear}(x) = \text{anchor\_adjustment}(x) + z_0 + leap_0$$

respectively.. Let us list down these similarities.

1) Both equations are the sum of 3 terms. Each of these terms corresponds with counterpart in the other method.

2) Both equations use memorization of an anchor year for the speedup. In simple method, the starting year of the decade serves as the anchor. In Conway's method, years with doomsyear value of zero are used as the anchor.

3) Both equations use z as the number of years between the input year and the anchor year. We can consider z as the offset from the anchor.

4) Both equations contain a leap count correction term that counts the number of leap years between the anchor year and the input year.

We now list down differences between the 2 equations and mention some advantages of our method over Conway's method.

1) In simple method, z is not computed. It is part of the input. In Conway's method, z_0 has to be calculated by subtracting the zero-anchor year from the input year.

2) In simple method, one has to memorize 10 digits for the anchoring. In Conway's method, one has to memorize 18 numbers for the anchoring.

3) In simple method, the leap count correction term follows a regular pattern for a given decade and is amenable to another speedup via memorization.

4) In simple method, we can always fall back to using the $2y + 10 (y \bmod 2)$ calculation if entries in the lookup table are forgotten.

Acknowledgements

I am very thankful to Head and Professor Dr Thidar Oo, all my colleague in Mathematics Department of Myingyan Degree College for their comments and oversee.

References

1. Berlekamp E. R., Conway J.H., and Guy R.K., (2004) Winning Ways for Your Mathematical Plays, Vol. 4, A. K. Peters, Wellesley, MA, Second Edition.
2. Carroll L., To Find the Day of the Week for Any Given Date, Nature , March 31 1887.
3. Gallian J. A., (2010) Contemporary Abstract Algebra, Brook/Cole, Cengage Learnign, Seventh Edition.
4. Limeback R., Doomsday Algorithm, http://rudy.ca/doomsday.html.
5. Richards E. G., (1998) Mapping Time: The Calendar and Its History, Oxford University Press, Oxford, U. K.
6. Rotman J. J., (1996) A First Course in Abstract Algebra, Prentice Hall, Upper Saddle River, NJ, Third Edition.

RELATIONS BETWEEN CAYLEY GRAPH AND VERTEX-TRANSITIVE GRAPH

Aye Aye Myint*

Abstract

In this paper, we first express basic concepts of graph theory. Then we define vertex-transitive graph and Cayley graph with a given group by using generating set or nongenerating set. Finally, we prove that every Cayley graph is vertex-transitive graph and we also give an example that the converse of this theorem is false.

Keywords: graph, digraph, connected, vertex-transitive, group, order, Cayley graph, Cayley digraph, diameter of a graph.

1. Basic Concepts of Graph Theory

A **graph** $G = (V(G), E(G))$ with n vertices and m edges consists of a vertex set $V(G)=\{ v_1, v_2, \ldots, v_n \}$ and an edge set $E(G) = \{ e_1, e_2, \ldots, e_m\}$, where each edge consists of two (possibly equal) vertices called its endpoints. We write uv for an edge $e = (u, v)$. If $uv \in E(G)$, then u and v are **adjacent**. The ends of an edge are said to be **incident** with the edge. A **loop** is an edge whose endpoints are equal. **Parallel edges** or **multiple edges** are edges that have the same pair of endpoints. A **simple graph** is a graph having no loops or multiple edges. A graph is **finite** if its vertex set and edge set are finite. We adopt the convention that every graph mentioned in this paper is finite, unless explicity constructed otherwise. The **degree** of a vertex v of a graph G is the number of edges of G which are incident with v. A graph is said to be **regular** (**k-regular**) if all its vertices have the same degree (k). A three-regular graph is also called a **cubic graph**. A simple graph in which each pair of distinct vertices is joined by an edge is called a **complete graph**. A complete graph on n vertices is denoted by K_n. A sequence of distinct edges of the form $v_0v_1, v_1v_2, \ldots, v_{r-1}v_r$ is called a **path of length r** from v_0 to v_r, denoted by (v_0, v_r)-path. The **distance** between two vertices u and v in a graph G is the length of the shortest path from u to v. The **diameter** of a graph is the maximum distance between two distinct vertices.

*. Dr., Lecturer, Department of Mathematics, Shwebo University

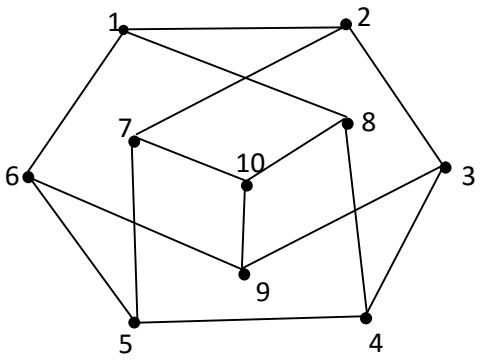


Figure 1.1: A graph G with diameter 2

A **subgraph** of a graph $G = (V(G), E(G))$ is a graph $Y = (V(Y), E(Y))$ with $V(Y) \subseteq V(G)$ and $E(Y) \subseteq E(G)$.Two vertices u and v of G are said to be **connected** if there is a (u, v)-path in G . A **connected graph** is a graph such that any two vertices are connected by a path, otherwise it is **disconnected**.

A **directed graph** (or **digraph**) $\vec{G} = (V(\vec{G}), E(\vec{G}))$ consists of a finite nonempty set $V(\vec{G})$, called **the set of vertices**, and set $E(\vec{G})$ of ordered pairs of (not necessarily distinct) vertices, called **the set of (directed) edges** or **arcs.** If $e = (u, v)$ or uv is a directed edge of $\vec{G}$, we say that e **joins** u to v, that u and v are **endpoints** of e(more specifically that u is the **tail** of e and v is the head of e). A digraph $\vec{G}$ is called **symmetric** if, whenever (u, v) is an arc of $\vec{G}$, then (v, u) is also. A digraph $\vec{G}$ is called **complete** if for every two distinct vertices u and v of $\vec{G}$, at least one of the arcs (u, v) and (v, u) is present in $\vec{G}$. A **complete symmetric digraph** of order n has both arcs (u, v) and (v, u) for every two distinct vertices u and v, denoted by $\vec{K}_n$.

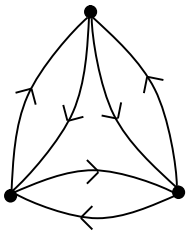


Figure 1.2: A complete symmetric digraph $\vec{K}_3$

2. Vertex-Transitive Graph

Before defining a vertex-transitive graph, we express the definition of an automorphism of a graph which plays a crucial role in determining the vertex-transitive graph.

An **isomorphism** from graph G to graph G' is a bijection $\phi: V(G) \to V(G')$ such that $uv \in E(G)$ if and only if $\phi(u)\ \phi(v) \in E(G')$. We say "G is isomorphic to G'", written $G \cong G'$, if there is an isomorphism from G to G'.

The graphs G and G' drawn below are isomorphic by an isomorphism that maps u, v, w, x, y, z to l, m, n, p, q, r respectively.

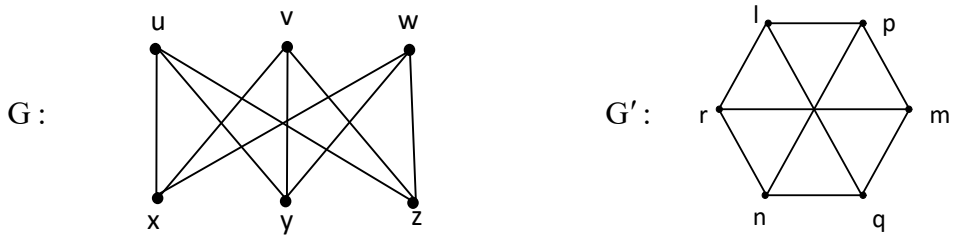


Figure 2.1: Isomorphic graphs G and G'

A **permutation** ϕ of V(G) is a function from V(G) into V(G) that is both one to one and onto.

An **automorphism** of a (simple) graph G is a permutation ϕ of V(G) which has the property that $uv \in E(G)$ if and only if $\phi(u)\ \phi(v) \in E(G)$, that is an isomorphism from G to G. The set of all automorphisms of a graph G forms a group under the operation of composition, which is called the **automorphism group**.

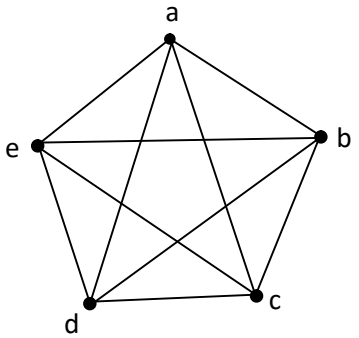


Figure 2.2: Complete graph K_5

In Figure 2.2, there is an automorphiosm ϕ of $V(K_5)$ such that $\phi(a) = b,\ \phi(b) = c,\ \phi(c) = d,\ \phi(d) = e,\ \phi(e) = a.$

For any two vertices u and v of G, there is an automorphism ϕ of G such that $\phi(u) = v,$ we say that G is **vertex-transitive.**

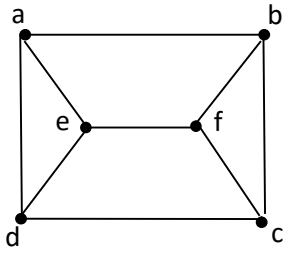


Figure 2.3: Vertex-transitive graph G

Now we interested in the structure of vertex-transitive graphs, in particular, Cayley graphs. First, we have to introduce some definitions of group theory.

3.Basic Definitions of Group Theory

A nonempty set of elements X is said to form a **group** if in X there is defined a binary operation, called the product and denoted by·, such that

(i) $a, b \in X$ implies that $a \cdot b \in X$ (closed).

(ii) $a, b, c \in X$ implies that $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (associative law).

(iii) There exists an element $e \in X$ such that $a \cdot e = e \cdot a = a$ for all $a \in X$ (the existence of an identity element in X).

(iv) For every $a \in X$ there exists an element $a^{-1} \in X$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$ (the existence of inverses in X).

We usually write a b instead of a ·b.

A **finite group** is a group which has a finite number of elements, otherwise we call it an **infinite group**.

The number of elements in X is called the **order** of X and it is denoted by O(X)or $|X|$. We define the **order of an element** x to be the least positive integer n such that $x^n = e$ and we denote it by O(x) or $|x|$.

A nonempty subset S of a group X is said to be a **subgroup** of X if, under the product in X, S itself forms a group.

If S is a subgroup of X, $a \in X$, then $aS = \{as | s \in S\}$. a S is called a **left coset** of S in X.

Let X be a group of permutation of a set A and $b \in A$, then the **stabilizer** of b (in X) is the subgroup $X_b = \{x \in X \mid x(b) = b\}$.

A group X is called a **cyclic group** if there exists an element $x \in$ X, such that every element of X can be expressed as a power of x. In that case x is called **generator** of X.

Let $D_n = \{x^i y^j \mid i = 0, 1;\ j = 0, 1, ..., n-1;\ x^2 = e = y^n, xy = y^{-1}x\}$. Then D_n is a group, called the **dihedral group**, $(n \geq 3)$. $O(D_n) = 2n$. In fact, we can write D_n also as

$$D_n = \{y, y^2, ..., y^{n-1}, y^n, xy, xy^2, ..., xy^{n-1}, x \mid x^2 = e = y^n, xy = y^{-1}x\}.$$

Let X be a group. A subset $H \subseteq X$ is a **generating set** of X if every element of X is obtainable as the product (or sum) of elements of H.

For the group Z_n, a nonempty set of integers modulo n is a generating set if and only if its greatest common divisor (gcd) is 1. For instance, the set {4,7} generates Z_{24}, since gcd (4,7) = 1 but {6,9} does not generate Z_{24}, since gcd (6,9) = 3.

If A is a finite set {1,2, . . . ,n}, then the group of all permutations of A is the **symmetric group on n letters**, and is denoted by S_n. Note that S_n has n! elements.

4. Cayley Graphs

Now we shall express the definitions of Cayley graphs and the constructions of Cayley graphs with their given groups.

4.1 Definitions

(i) Let X be a group and H a subset of X not containing the identity e. Then the **Cayley digraph** $\vec{C}$ has vertex set $V(\vec{C}) = X$ and arc set $E(\vec{C}) = \{(g, gh) \mid h \in H,\ g \in X\}$. We write $\vec{C} = \vec{C}(X, H)$.

(ii) Let $H = X - e$. Then the resulting Cayley digraph will be denoted by $\vec{K} = \vec{K}(X, H)$ and called the **complete Cayley digraph**.

(iii) Let X be a group and H a subset of X not containing the identity e such that $h \in H$ implies $h^{-1} \in H$ (that is, $H = H^{-1}$, where $H^{-1} = \{h^{-1} \mid h \in H\}$). Then the graph with vertex set $V(C) = X$ and edge set $E(C) = \{(g, gh) \mid h \in H, g \in X\}$ is called the **Cayley graph** C corresponding to X, H. We write $C = C(X, H)$. Equivalently, the Cayley graph $C = C(X, H)$ is the simple graph whose vertex set and edge set are defined as follows:

$$V(C) = X;\ \ E(C) = \{(g, h) \mid g^{-1}h \in H, \text{ where } g \in X,\ h \in H\}.$$

(iv) When H is a set of generators for X the Cayley digraph and the Cayley graph will be referred to as the **basic Cayley digraph** and the **basic Cayley graph** respectively.

4.2 Examples

(i) Let X be the group Z_5, the set of integers modulo 5.

Let H be generating set $\{1\}$.

We can construct the Cayley digraph $\vec{C} = \vec{C}(Z_5, H)$ which has the vertex set $V(\vec{C}) = Z_5 = \{0, 1, 2, 3, 4\}$; and the arc set $E(\vec{C}) = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)\}$.

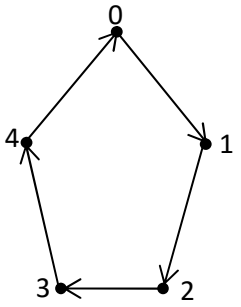


Figure 4.1: The Cayley digraph $\vec{C}(Z_5, \{1\})$

(ii) Let X be the cyclic group C_4 generated by a and $H = \{a, a^2, a^3\}$.

We can construct the complete Cayley digraph $\vec{K} = \vec{K}(C_4, H)$ which has the vertex set $V(\vec{K}) = C_4 = \{1, a, a^2, a^3\}$; and the arc set

$$E(\vec{K}) = \{(1,a),(1,a^2),(1,a^3),(a,a^2),(a,a^3),(a,1),(a^2,a^3),(a^2,1),(a^2,a),(a^3,1),(a^3,a),(a^3,a^2)\}.$$

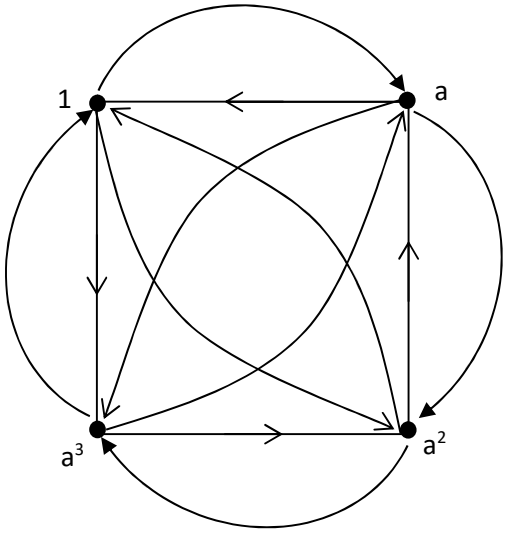


Figure4.2: The complete Cayley digraph $\vec{K}$ $(C_4, \{a, a^2, a^3\})$

(iii) Let X be the symmetric group $S_3 = \{1, (12), (13), (23), (123), (132)\}$.

Let $H = \{(12), (13), (23)\}$.

Then $H^{-1} = H$. We can construct the Cayley graph $C = C(S_3, H)$ which has

$$V(C) = S_3; \quad E(C) = \{(1,(12)), (1,(13)), (1,(23)), ((12), (132)), ((12), (123)),$$
$$((13), (132)), ((13), (123)), ((23), (132)), ((23), (123))\}.$$

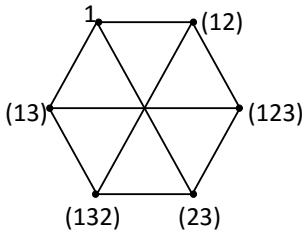


Figure4.3: The Cayley graph $C(S_3, \{(12), (13), (23)\})$.

4.3Theorem

The Cayley graph $C(X, H)$ is well-defined and is connected if and only if H is a set of generators for X.

Proof. See [7].

4.4 Examples

(i) Let X be the dihedral group $D_4 = \{1, r, r^2, r^3, s, rs, r^2s, r^3s\}$, where $r^4 = s^2 = 1$, $sr = r^{-1}s$ and $H = \{ r, s \}$.

Then H is a generating set for D_4. We can construct the Cayley digraph $\vec{C} = \vec{C}(D_4, H)$ has the vertex set $V(\vec{C}) = D_4$ and the arc set $E(\vec{C}) = \{ (g, gh) \mid g \in D_4, h \in H \}$.

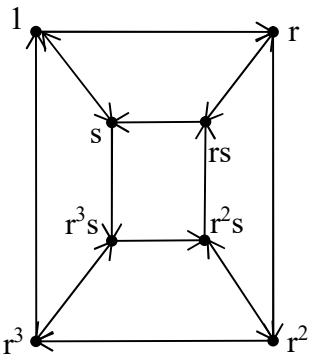


Figure 4.4: The connected Cayley digraph $\vec{C}(D_4, \{r, s\})$

(ii) Let X be $Z_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$ and let $H = \{2, 6\}$.

Since $H = H^{-1}$ and H is not a generating set, we can construct the disconnected Cayley graph C has vertex set $V(C) = Z_8$ and edge set $E(C) = \{ (g, gh) \mid g \in Z_8, h \in H \}$.

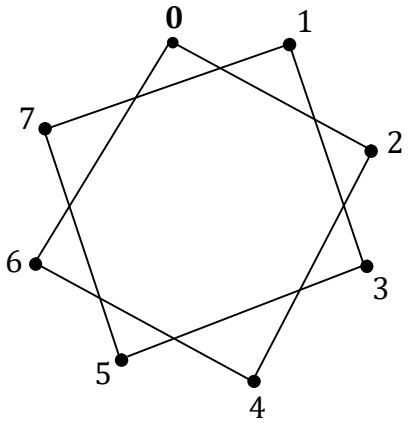


Figure 4.5: The disconnected Cayley graph $C(Z_8, H)$

In the above examples, we see that if the subset H is a generating set for the given group then the Cayley digraph is connected and if H is not a generating set then the Cayley graph is disconnected.

5. Relations between Cayley Graph and Vertex-Transitive Graph

In this section, we interested in a relation between Cayley graph and vertex-transitive graph.

5.1Theorem

Every Cayley graph $C(X, H)$ is vertex-transitive.

Proof.

For each g in X we define a permutation ϕ_g of $V(C) = X$ by the rule $\phi_g(h) = gh,\ h \in X.$

This permutation ϕ_g is an automorphism of C , for

$$(h,k) \in E(C) \Rightarrow\ h^{-1}k \in H$$

$$\Rightarrow\ (gh)^{-1}(gk) \in H$$

$$\Rightarrow\ \left(\phi_g(h), \phi_g(k)\right) \in E(C)$$

Now for any $h, k \in X,\ \phi_{kh^{-1}}(h) = \left(kh^{-1}\right)h = k.$

Hence Cayley graph $C(X, H)$ is vertex-transitive.

5.2 Petersen graph

The **Petersen graph** P(5,2) is a cubic graph having a vertex set $V = \{u_0,...,u_4, v_0,...,v_4\}$ and an edge set $E = \{(u_i,u_{i+1}),(u_i,v_i),(v_i,v_{i+2}) \mid i = 0,...,4\}$ where all the subscripts are taken modulo 5. The **generalized Petersen graph** $P(n, k)(n \geq 5,\ 0 < k < n)$ is the cubic graph having a vertex set $\{u_0,...,u_{n-1}, v_0,...,v_{n-1}\}$ and an edge set $\{(u_i,u_{i+1}),(u_i,v_i),(v_i,v_{i+2}) \mid i = 0,...,n-1\}$ where all the subscripts are taken modulo n.

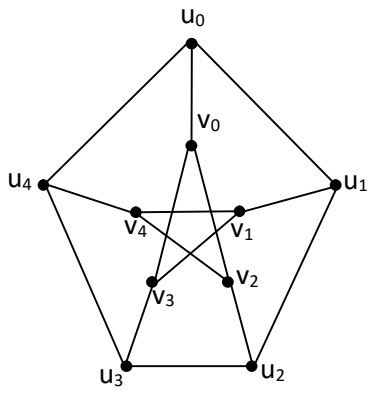


Figure5.1: The Petersen graph P(5, 2).

The following is an example of a vertex-transitive graph which is not a Cayley graph.

5.3 Example

The Petersen graph is vertex-transitive but it is not a Cayley graph.

Indeed, we can see the diameter of the Petersen graph is 2 and the diameter of a Cayley graph $C = C(X, H)$ is the smallest positive integer n such that $X = H \cup H^2 \cup ... \cup H^n$ where $H^2 = \{hk \mid h, k \in H\}$ and $H^i = H^{i-1}H$ for $i \geq 3$.

We now show that all the Cayley graphs of order 10 having degree 3 are of diameter greater than 2 and so none of them is the Petersen graph.

There are two groups of order 10. The first one is the cyclic group Z_{10} and the second one is the dihedral group D_5. The group operation here are additions and we replace H^{-1} by – H.

Case 1.

$X = Z_{10} = \{0, 1, ..., 9\}$.

Since – H = H and $|H| = 3$, $5 \in H$ and H can only be one of the following four sets

$H_1 = \{1, 5, 9\}$, $H_2 = \{2, 5, 8\}$,

$H_3 = \{3, 5, 7\}$, $H_4 = \{4, 5, 6\}$.

Now $|H_i + H_i| = 5$ for each $i = 1, 2, 3, 4$.

Thus the diameter of C is greater than 2.

Case 2.

$X = D_5 = \{0, b, 2b, 3b, 4b, a, a+b, a+2b, a+3b, a+4b\}$ where $2a = 0$, $5b = 0$ and $b + a = a + 4b$.

In this case a, a + b, a + 2b, a + 3b and a + 4b are the only elements of order 2 in X.

Hence H can only be one of the following three types of sets

$H_1 = \{a + jb, b, 4b\}$, $j = 0, 1, 2, 3, 4$;

$H_2 = \{a + jb, 2b, 3b\}$, $j = 0, 1, 2, 3, 4$;

$H_3 = \{a + j_1b, a + j_2b, a + j_3b\}$, $0 \le j_1 < j_2 < j_3 \le 4$.

Now $|H_i + H_i| = 5$ for each $i = 1, 2, 3$.

Thus the diameter of C is greater than 2 also.

Petersen graph is a vertex-transitive graph but it is not a Cayley graph.

From the above example, we see that every vertex-transitive graph is not a Cayley graph. But every vertex-transitive graph can be constructed almost like a Cayley graph. This result will be shown in Theorem 5.5. We shall apply the following theorem to prove Theorem 5.5.

5.4 Theorem

Let S be a subgroup of a finite group X and let H be a subset of X such that $H^{-1} = H$ and $H \cap S = \varnothing$. If G is the graph having vertex set $V(G) = X/S$

(the set of all left cosets of S in X) and edge set $E(G) = \{(xS, yS) \mid x^{-1}y \in SHS\}$, then G is vertex-transitive.

Proof.

We first show that the graph G is well-defined.

Suppose that $(xS, yS) \in E(G)$ and $x_1S = xS,\ y_1S = yS.$

Then $x_1 = xs,\ y_1 = yk$ for some $s, k \in S.$

$$\text{Now} \quad x^{-1}y \in SHS \Rightarrow (xs)^{-1}(yk) \in SHS$$

$$\Rightarrow x_1^{-1}y_1 \in SHS$$

$$\Rightarrow (x_1s,\ y_1s) \in E(G).$$

Hence the graph G is well-defined.

Next, for each $g \in X$ we defined a permutation ϕ_g of $V(G) = X/S$ by the rule such that $\phi_g(xS) = gxS, \quad xS \in X/S.$

This permutation ϕ_g is an automorphism of G, for

$$(xS,\ yS) \in E(G) \quad \Rightarrow x^{-1}y \in SHS$$

$$\Rightarrow (gx)^{-1}(gy) \in SHS$$

$$\Rightarrow (gxS,\ gyS) \in E(G)$$

$$\Rightarrow \left(\phi_g(xS),\ \phi_g(yS)\right) \in E(G).$$

Finally, for any $xS,\ yS \in X/S,\ \phi_{yx^{-1}}(xS) = yx^{-1}(xS) = yS.$

Hence the graph G is vertex-transitive.

The graph G constructed in above theorem is called the **group-coset graph** X/S generated by H and is denoted by $G(X/S, H).$

5.5 Theorem

Let G be a vertex-transitive graph whose automorphism group is A. Let $H = A_b$ be stabilizer of $b \in V(G)$. Then G is isomorphic with the group-coset graph $G(A/H, S)$ where S is the set of all automorphism x of G such that $(b, x(b)) \in E(G)$.

Proof.

We can see that $S^{-1} = S$ and $S \cap H = \varnothing$.

We now show that $\phi: A/H \to G$ given by $\phi(xH) = x(b)$, where $xH \in A/H$, defines a map.

Suppose $xH = yH$.

Then $y = xh$ for some $y \in H$.

$\phi(yH) = y(b) = (xh)(b) = x(h(b)) = x(b) = \phi(xH)$.

We next show that ϕ is a graph isomorphism.

Suppose $\phi(xH) = \phi(yH)$.

Then $x(b) = y(b)$

$y^{-1}x(b) = b$

$y^{-1}x \in H$

$x \in yH$

$yH = xH$.

So ϕ is one to one.

Let c be a vertex of G.

Since G is vertex-transitive, there exists z in A such that $z(b) = c$

Thus $\phi(zH) = z(b) = c$.

So ϕ is onto.

$$\text{Next} \quad (xH,\ yH) \in E\big(G(A/H,S)\big) \quad \Leftrightarrow x^{-1}y \in HSH$$

$$\Leftrightarrow x^{-1}y = hzk \text{ for some } h,k \in H,\ z \in S$$

$$\Leftrightarrow h^{-1}x^{-1}yk^{-1} = z$$

$$\Leftrightarrow \big(b,\ h^{-1}x^{-1}yk^{-1}(b)\big) \in E(G)$$

$$\Leftrightarrow \big(b,\ x^{-1}y(b)\big) \in E(G)$$

$$\Leftrightarrow \big(x(b),\ y(b)\big) \in E(G)$$

$$\Leftrightarrow \big(\phi(xH),\ \phi(yH)\big) \in E(G).$$

Thus G is isomorphic with the group-coset graph $G(A/H, S)$.

References

1. Alspach, B., "*CS E6204 Lecture 6 Cayley Graphs*", University of Regina, Canada, pp. 2-15,(2010).
2. Biggs, N., "Algebraic Graph Theory", Cambridge University Press, Cambridge, (1993).
3. Bondy, J.A. and Murty, U.S.R., "Graph Theory with Applications", The Macmillan Press Ltd, London, (1976).
4. Chen, T. Y., "*Connectivity in Vertex-Transitive Graphs*", M.Sc Thesis Submitted to Simon Fraser University, Canada, pp.27-38, (1994).
5. Herstein, I.N., "Topics in Algebra", John Wiley and Sons, Inc., New York, (1975).
6. Khanna, V. K. and Bhambri, S.K., "A Course in Algebra", Vikas Publishing House Pvt.,Ltd., New Delhi, (1998).
7. Parthasarathy, K.R., "Basic Graph Theory", Tata McGraw-Hill Publishing Company Limited, New Delhi, (1994).
8. Yap, H.P., "Some Topics in Graph Theory", Cambridge University Press, Cambridge, 1986.

GRAPHS WITH THE SPECIFIED EDGE GEODETIC NUMBERS

Kyaw Lin Aung[1], Tin Mar Htwe[2]

Abstract

Firstly we state some properties of the edge geodetic number of the connected graphs. Then the edge geodetic numbers of some special graphs are derived. Next we study the graphs with the edge geodetic number 2. We also state the necessary and sufficient conditions for a graph G with n vertices to have the edge geodetic number $g_e(G) = n - 1$ or $g_e(G) = n$. Finally we characterize the graphs which have the specified edge geodetic numbers.

Keywords: Edge geodetic cover, Edge geodetic basis, Edge geodetic number

1. Some Properties of the Edge Geodetic Number of a Connected Graph

In this section, being based on [1] through [4], we state the following basic results of the edge geodetic number of a connected graph.

1.1 Definitions.

An ***edge geodetic cover*** of a graph G is a set $S \subseteq V$ where V is the set of vertices of G such that every edge of G is contained in a geodesic joining some pair of vertices in S. The ***edge geodetic number*** $g_e(G)$ of G is the minimum order of its geodetic covers, and any edge geodetic cover of order $g_e(G)$ is an ***edge geodetic basis***. If S is an edge geodetic basis of G and a vertex $x \in S$, then x is called a ***basic vertex*** with respect to the basis S.

1.2 Example.

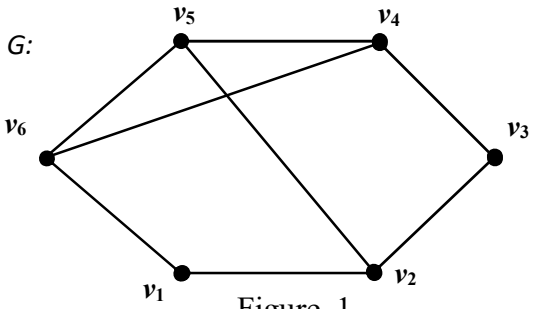


Figure. 1

[1.] Assistant Lecturer, Department of Mathematics, Sittway University,
[2.] Professor(Head), Department of Mathematics, Taungoo University

Consider the graph G shown in Fig. 1, any vertex set S containing two vertices from $\{ v_1, v_2, v_3, v_4, v_5, v_6\}$ is not an edge geodetic cover. Thus, $g_e(G) \geq 3$.

If $S = \{ v_1, v_3, v_5\}$, we can see that it is not an edge geodetic cover of G. But if $S = \{ v_2, v_4, v_6\}$, it is an edge geodetic cover of G and has minimum order. Thus the edge geodetic number $g_e(G)$ of G is 3 and S is an edge geodetic basis of G and v_2, v_4, v_6 are basic vertices with respect to S.

1.3 Remark. The edge geodetic number of a disconnected graph is the sum of the edge geodetic number of its components. Thus we will consider only connected graphs in the next sections.

1.4 Theorem. For every nontrivial graph G of order n, $2 \leq g_e(G) \leq n$.

Proof. An edge geodetic cover needs at least two vertices and therefore $g_e(G) \geq 2$. Clearly the set of all vertices of G is an edge geodetic cover of G. This means that $g_e(G) \leq n$. Thus $2 \leq g_e(G) \leq n$.

1.5 Theorem. Each extreme vertex of G belongs to every edge geodetic basis of G.

Proof. Let x be an extreme vertex of a connected graph G.

Suppose $W \subseteq V$ is any edge geodetic basis of G and $x \notin W$.

Then there are two vertices u_1, v_1 in W such that the shortest path between them must contain x, say P: $u_1 \ldots uxv \ldots v_1$.

Since x is an extreme vertex, u and v are adjacent. Moreover, the other path Q: $u_1 \ldots uv \ldots v_1$ exists between u_1 and v_1 having length, one less than the path P, and will not contain the edges ux and xv.

It contradicts the facts that P is the shortest path and W is an edge geodetic basis of G.

Hence $x \in W$.

1.6 Corollary. Each pendant vertex of G belongs to every vertex geodetic basis of G.

Proof. Every pendent vertex is an extreme vertex.

It completes the proof.

1.7 Theorem. For any graph, no cut vertex of G belongs to any edge geodetic basis of G.

Proof. Let x be a cut vertex of a connected graph G and $W \subseteq V$ be an edge geodetic basis of G.

Suppose $x \in W$.

Since x is a cut vertex, $G - x$ is disconnected and suppose that $G - x$ consists of k components $G_1, G_2, \ldots, G_k$ where $k \geq 2$.

Obviously W contains at least one vertex y_i (being adjacent to x) of each component G_i, $i = 1, 2, \ldots, k$.

Otherwise, W will not be the edge geodetic cover of G.

Consider the set $W \setminus \{x\}$ and any edge $u_1u_2 \in E$ where $u_1 \neq x$, $u_2 \neq x$.

Then u_1u_2 is on a shortest vw-path in G where $v, w \in W$. If $v \neq x$ and $w \neq x$, then u_1u_2 lies on a shortest vw-path where $v, w \in W \setminus \{x\}$.

Suppose $v = x$ and $w = y_i$ where $y_i \in G_i$. Then u_1u_2 is on a shortest xy_i-path, say P.

Consider the vertex y_j of G_j where $j \neq i$. Let Q be a shortest y_jx-path.

Then the union of P and Q is a shortest y_iy_j-path say R, both xu_1 and $u_1u_2$lie on R. It follows from these discussions that $W \setminus \{x\}$ is an edge geodetic cover of G.

This contradicts to the hypothesis that W is an edge geodetic basis of G.

Hence $x \notin W$.

The proofs of Theorem 1.5, Corollary 1.6 and Theorem 1.7 can also be seen in [4].

From Theorem 1.4, Theorem 1.5 and Theorem 1.7, we obtained the following theorem which states the bounds of edge geodetic numbers of a graph.

1.8 Theorem. For any graph G of order n with m extreme vertices and k cut vertices, $\max\{2, m\} \leq g_e(G) \leq n - k$.

By applying above theorems, we derive the edge geodetic numbers of some well-known graphs that are described in Theorem 1.9 to Theorem 1.13.

1.9 Theorem. The edge geodetic number $g_e(P_n)$ of any path $P_n$is 2.

Proof. Every path has two extreme vertices. Thus the number of extreme vertices of the path P_n is 2.

By Theorem 1.8, $g_e(P_n) \geq \max\{2, 2\}$

$= 2.$

Moreover the path P_n has $n - 2$ cut vertices.

By Theorem 1.8, $g_e(P_n) \leq n - (n - 2)$

$= 2.$

Hence, $g_e(P_n) = 2$.

1.10 Theorem. The edge geodetic number of any tree is the number of its pendant vertices.

Proof. Consider the tree T with n vertices and m pendant vertices. Every tree has at least two pendant vertices and every pendant vertex is extreme vertex. Thus $m \geq 2$.

By Theorem 1.8, $g_e(T) \geq \max\{2, m\}$

$= m.$

Moreover, every vertex in a tree is either pendant vertex or cut vertex.

Hence, the number of cut vertices of the tree T is $n - m$. By Theorem 1.8, $g_e(T) \leq n - (n - m)$

$= m.$

So, we get $g_e(T) = m$.

1.11 Theorem. The edge geodetic number of an even cycle is 2 and of an odd cycle is 3.

Proof. Let C be a cycle with $2n$ vertices $v_1, v_2, \ldots, v_{2n}$ in order.

Let $W = \{ v_1, v_{n+1}\}$. Then every edge of C is contained in a shortest path joining the vertices of W. Therefore W is an edge geodetic cover of C and so $g_e(C) \leq |W| = 2$. Since $g_e(C) \geq 2$, $g_e(C) = 2$.

Let C be a cycle with $2n + 1$ vertices $v_1, v_2, \ldots, v_{2n+1}$ in order.

Let $W = \{ v_1, v_{n+1}, v_{n+2}\}$. Then every edge of C is contained in a shortest path joining the vertices of W. Therefore W is an edge geodetic cover of C and so $g_e(C) \leq |W| = 3$. Since $g_e(C) \geq 2$, $g_e(C) = 3$.

1.12 Theorem. For the complete graph $K_n (n \geq 2)$, $g_e(K_n) = n$.

Proof. Consider the complete graph K_n.

In any complete graph every vertex is extreme vertex and no vertex is cut vertex. By Theorem 3.11, $g_e(K_n) \geq \max \{ n, 2\}$

$$= n.$$

And $g_e(K_n) \leq n - 0$

Hence, $g_e(K_n) = n$.

1.13 Theorem. For the complete bipartite graph $G = K_{m,n}$,

$g_e(G) = 2$ if $m = n = 1$;

$g_e(G) = n$ if $m = 1, n \geq 2$;

$g_e(G) = \min \{m, n\}$ if $m, n \geq 2$.

Proof. (i) For $m = n = 1$, $K_{1,1}$is a path P_2 and hence $g_e(K_{1,1}) = 2$.

(ii) For $m = 1$ and $n \geq 2$, consider $K_{1,n}$.

$K_{1,n}$ has n extreme vertices and only one cut vertex.

By Theorem 3.11, $\max \{n, 2\} \leq g_e(K_{1,n}) \leq (n + 1) - 1$

$$n \leq g_e(K_{1,n}) \leq n$$

$$g_e(K_{1,n}) = n.$$

(iii) Now consider $K_{m,n}$for m, $n \geq 2$ and suppose $m \leq n$.

The $K_{m,n}$ can be decomposed as follows:

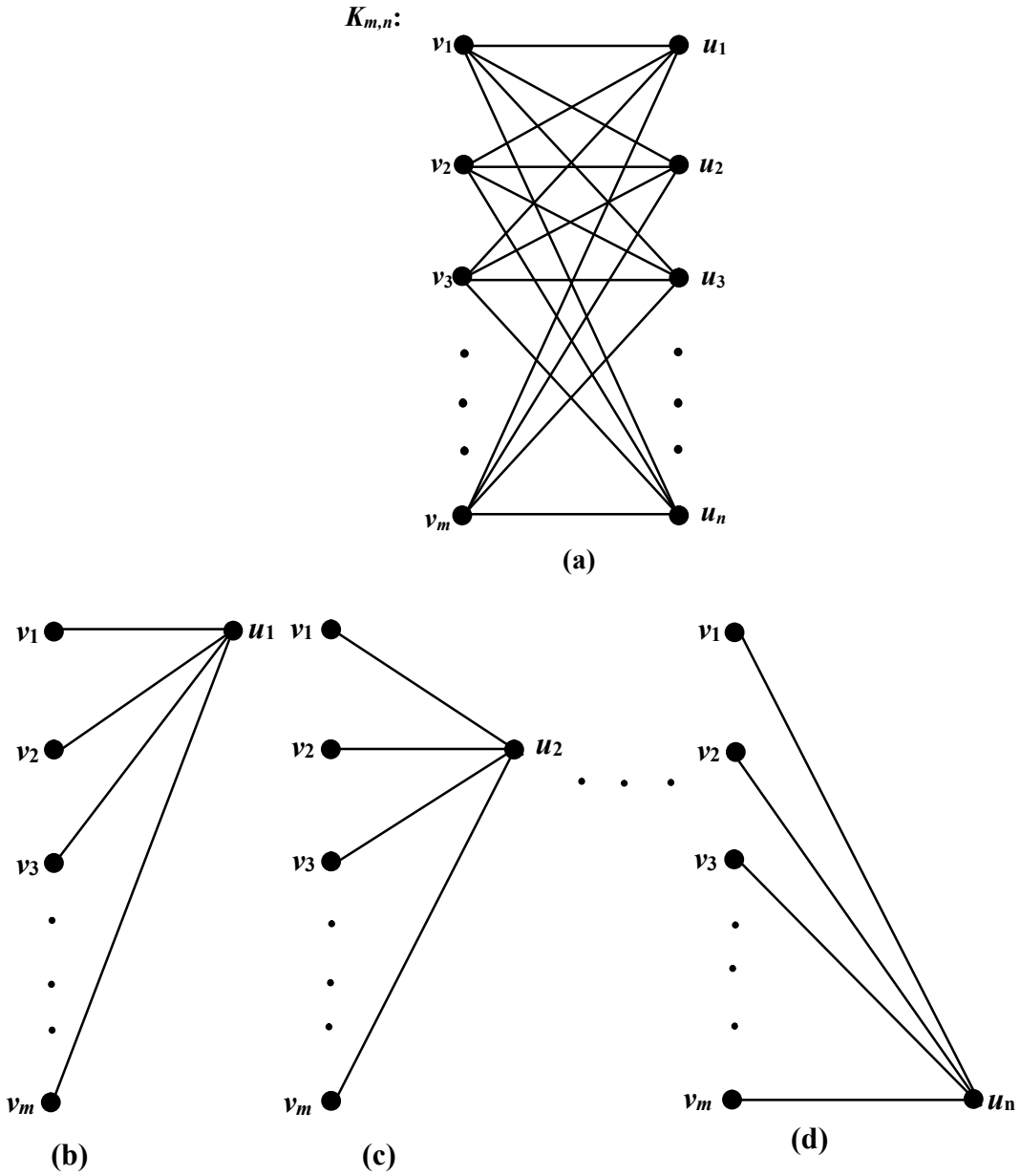


Figure. 2

Now, $g_e(K_{m,n}) = g_e(\bigcup_n K_{m,1})$

= minimum order of edge cover of $\bigcup_n K_{m,1}$

= minimum order of union of edge geodetic cover of $K_{m,1}$

= minimum order of $\bigcup_n \{ v_1, v_2, \ldots, v_m \}$

= minimum order of $\{ v_1, v_2, \ldots, v_m \}$

$= m$.

The proofs of theorems from Theorem 1.10 to Theorem 1.13 can also be seen in [4].

2. Conditions for Graphs to Have Some Specified Edge Geodetic Numbers

In this section, we study some theorems that characterize graphs for which the edge geodetic number $g_e(G)$ is 2and some necessary and sufficient conditions of a graph G with n vertices to have the edge geodetic number $g_e(G) = n - 1$ org$_e(G) = n$. We mainly refer to [5].

2.1 Theorem. For a connected graph G, $g_e(G) = 2$ if and only if there exist peripheral vertices u and v such that every edge of G is on a diametral path joining u and v.

2.2 Theorem. If G has exactly one vertex v of degree $n - 1$, then $g_e(G) = n - 1$.

2.3 Corollary. If G has exactly one vertex v of degree $n - 1$, then G has a unique edge geodetic basis consisting of all the vertices of G other than v.

Proof. Let G be a graph with n vertices and suppose the vertex v is the only one vertex of degree $n - 1$. By Theorem 2.2,$g_e(G) = n - 1$.

Since $d(v) = n - 1$, v must be adjacent to the remaining $v_1, v_2, \ldots, v_{n-1}$ vertices of G. It means that v is on the shortest path with length 2 of any two vertices of $S = \{ v_1, v_2, \ldots, v_{n-1}\}$. In other words, all the edges of G which join v with each of $v_1, v_2, \ldots, v_{n-1}$ are on the shortest path of any two vertices of S.

On the other hand, each edge joining any two vertices from v_1, v_2, . . .,v_{n-1}does not lie on any geodesic joining two vertices of S other than themselves. Thus S is a geodetic cover and $g_e(G) \leq n-1$. But $g_e(G) = n-1$ and S is the geodetic basis of G. Since v is the exactly only vertex of degree $n-1$, S is a unique edge geodetic basis consisting of all the vertices of G other than v.

2.4 Theorem. Let G be a graph of order $n \geq 3$. If G contains a cut vertex of degree $n-1$, then $g_e(G) = n-1$.

Proof. Let v be a cut vertex of G of degree $n-1$. It must be the only such vertex. For, suppose u be another cut vertex of degree $n-1$. So, u will be adjacent with the remaining $n-1$ vertices of G. Although we remove v from G, G will bestill connected. It contradicts that v is a cut vertex. Sov is the only vertex of degree $n-1$ and hence by Theorem 2.3, $g_e(G) = n-1$.

Now we discuss the edge geodetic number of a graph having more than one vertex of degree $n-1$.

2.5 Theorem. If G has more than one vertex of degree $n-1$, then every edge geodetic cover contains all those vertices of degree $n-1$.

2.6 Theorem. For any graph G with at least two vertices of degree $n-1$, $g_e(G) = n$.

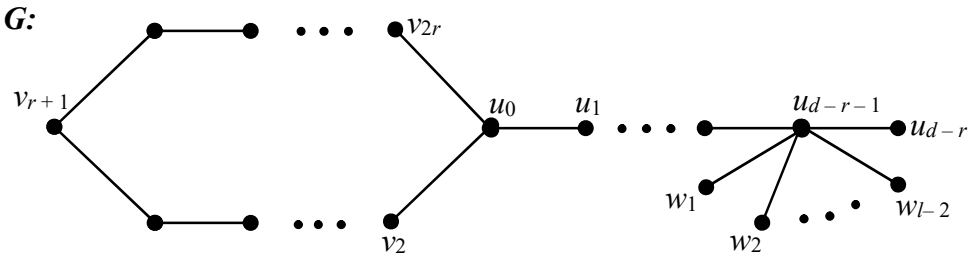
2.7 Theorem. For positive integers r, d and $l \geq 2$ with $r < d \leq 2r$, there exists a connected graph G with rad $G = r$, diam $G = d$, $g_e(G) = l$.

3. Constructions of Graphs with the Specified Edge Geodetic Numbers

From the previous sections, we can construct some graphs which have the specified edge geodetic numbers.

3.1 Graphs with the given the edge geodetic numbers

For any positive integer $l \geq 2$, the graphs with the $(d + r + l - 2)$ vertices have the edge geodetic number $g_e(G) = l$ where $r \leq d \leq 2r$ is as follows:



Figuer. 1

3.2 Graphs with the edge geodetic number 2

(1) For the edge geodetic number $g_e(G) = 2$, the graph is stated below.

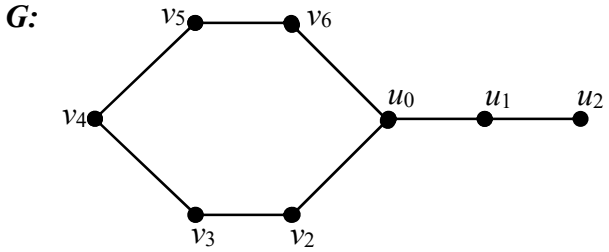


Figure. 2

(2) For the pathP_n with n vertices, the edge geodetic number is $g_e(P_n) = 2$.

Figure. 3

(3) For the even cycle $C_{2n}$with $2n$ vertices, the edge geodetic number is $g_e(C_{2n}) = 2$.

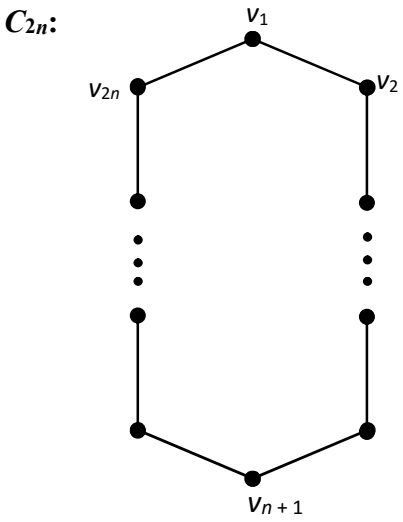


Figure. 4

(4) The cube Q_n with n vertices has the edge geodetic number is $g_e(Q_n) = 2$.

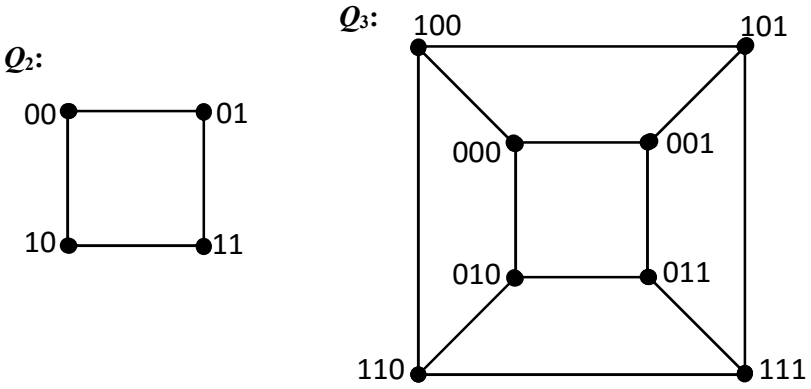


Figure. 5

3.3 Graphs with *n* vertices having the edge geodetic number*n* – 1

(1) The star with *n* vertices, say $K_{1,n-1}$ has the edge geodetic number $n - 1$.

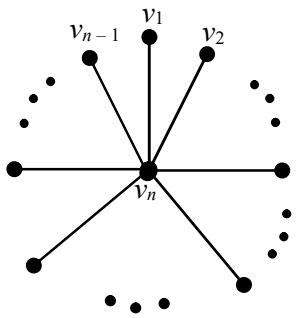


Figure. 6

(2) The wheel with *n* vertices, say $W_{1,n-1}$ has the edge geodetic number $n - 1$.

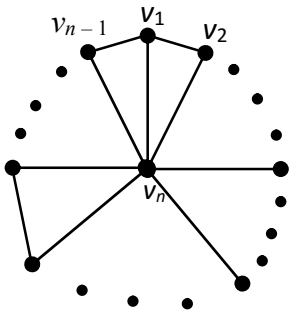


Figure. 7

3.4 Graphs with *n* vertices having the edge geodetic number *n*

(1) The complete graph, K_n has the edge geodetic number n.

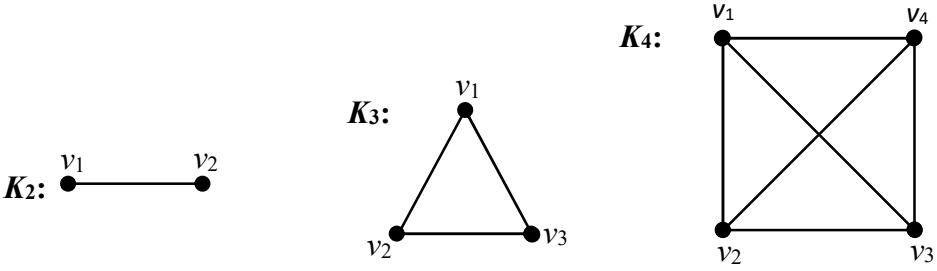


Figure. 8

References

1. M. Atici, On the edge geodetic number of a graph, *Intern. J. Computer Math*., **80**, 7 (2003), 853-861.
2. J. A. Bondy and U.S.R. Murty, *Graph Theory with Applications,* Fifth Printing, American Elsevier, New York, 1982.
3. F. Buckley and F. Harary, *Distance in Graphs*, Addison Wesley, New York, 1990.
4. A. P. Santhakumaran and J. John, Edge geodetic number of a graph, *Journal of Discrete Mathematical Sciences and Cryptography*, **10**,3 (2007), 415-432.

m^{th} LEVEL HARMONIC NUMBERS

Aung Phone Maw[1] and Aung Kyaw[2]

Abstract

We define m^{th} level harmonic numbers as a generalization of harmonic numbers. Then we construct the table of m^{th} level harmonic numbers which is like the Pascal's triangle. A formula for m^{th} level harmonic numbers containing binomial coefficients, as a generalization of Euler's formula for harmonic numbers, is also presented. From this formula, we also derive some relations between harmonic numbers and binomial coefficient.

m^{th} Level Harmonic Numbers

For a positive integer n, a harmonic number H_n is defined as $H_n = \sum_{k=1}^{n} \frac{1}{k}$. Here we define m^{th} level harmonic number as follows:

$$H_n^{(0)} = 1;\ H_n^{(m)} = \sum_{k=1}^{n} \frac{1}{k} H_k^{(m-1)} \quad \text{for any positive integer } m.$$

Since $H_n = \sum_{k=1}^{n} \frac{1}{k} = \sum_{k=1}^{n} \frac{1}{k} \cdot 1 = \sum_{k=1}^{n} \frac{1}{k} H_k^{(0)} = H_n^{(1)}$, one can see that m^{th} level harmonic number is a generalization of a harmonic number.

Table of m^{th} Level Harmonic Numbers

From the definition of m^{th} level harmonic number, $H_n^{(0)} = 1$ and $H_1^{(m)} = 1$. For every $n \geq 2$ we have

1. First year student, Department of Mathematics, University of Yangon
2. Professor, Department of Mathematics, University of Yangon

$$H_n^{(m)} = \sum_{k=1}^{n} \frac{1}{k} H_k^{(m-1)}$$

$$= \sum_{k=1}^{n-1} \frac{1}{k} H_k^{(m-1)} + \frac{1}{n} H_n^{(m-1)}$$

$$H_n^{(m)} = H_{n-1}^{(m)} + \frac{1}{n} H_n^{(m-1)}$$

From these facts we can construct the table of m^{th} level harmonic numbers like Pascal's triangle as follows:

n \ m	0	1	2	3	4
1	1	1	1	1	1
2	1	$\frac{3}{2}$	$\frac{7}{4}$	$\frac{15}{8}$	$\frac{31}{16}$
3	1	$\frac{11}{6}$	$\frac{85}{36}$	$\frac{575}{216}$	$\frac{1387}{491}$
4	1	$\frac{25}{12}$	$\frac{415}{144}$	$\frac{5845}{1728}$	$\frac{12456839}{3393792}$

In the above table, $H_n^{(m)}$ can be calculated as $\begin{matrix} H_{n-1}^{(m)} \\ \downarrow \\ + \\ H_n^{(m-1)} \xrightarrow{\times \frac{1}{n}} H_n^{(m)} \end{matrix}$.

m^{th} Level Harmonic Numbers and Binomial Coefficients

Euler's formula for harmonic numbers containing binomial coefficients is

$$H_n = H_n^{(1)} = \sum_{k=1}^{n} (-1)^{k+1} \frac{1}{k} \binom{n}{k}.$$

We will show that

$$H_n^{(m)} = \sum_{k=1}^{n} (-1)^{k+1} \frac{1}{k^m} \binom{n}{k},$$

this formula can be seen as a generalization of Euler's formula for harmonic numbers.

Proof. We will prove by induction.

When $n=1,\ H_1^{(m)}=1=\sum_{k=1}^{1}(-1)^{k+1}\frac{1}{k^m}\binom{1}{k}$.

When $m=0,\ H_n^{(0)}=1$ and

$$\sum_{k=1}^{n}(-1)^{k+1}\frac{1}{k^0}\binom{n}{k}=\binom{n}{1}-\binom{n}{2}+\binom{n}{3}-\binom{n}{4}+\cdots+(-1)^{n+1}\binom{n}{n}=1.$$

Therefore $H_n^{(0)}=\sum_{k=1}^{n}(-1)^{k+1}\frac{1}{k^0}\binom{n}{k}$.

Now we will show that the formula is true for $H_n^{(m)},\ n\geq 2, m\geq 1$, by assuming that the formula is true for $H_{n-1}^{(m)}$ and $H_n^{(m-1)}$. Since $H_n^{(m)}=H_{n-1}^{(m)}+\frac{1}{n}H_n^{(m-1)}$, then

$$\begin{aligned}
H_n^{(m)} &= H_{n-1}^{(m)}+\frac{1}{n}H_n^{(m-1)} \\
&= \sum_{k=1}^{n-1}(-1)^{k+1}\frac{1}{k^m}\binom{n-1}{k}+\frac{1}{n}\sum_{k=1}^{n}(-1)^{k+1}\frac{1}{k^{m-1}}\binom{n}{k} \\
&= \sum_{k=1}^{n-1}(-1)^{k+1}\frac{1}{k^m}\frac{n-k}{n}\binom{n}{k}+(-1)^{n+1}\frac{1}{n\cdot n^{m-1}}\cdot 1+\sum_{k=1}^{n-1}(-1)^{k+1}\frac{1}{n\cdot k^{m-1}}\binom{n}{k}+ \\
&= \sum_{k=1}^{n-1}(-1)^{k+1}\frac{1}{k^m}\frac{(n-k)+k}{n}\binom{n}{k}+(-1)^{n+1}\frac{1}{n^m} \\
&= \sum_{k=1}^{n-1}(-1)^{k+1}\frac{1}{k^m}\binom{n}{k}+(-1)^{n+1}\frac{1}{n^m} \\
H_n^{(m)} &= \sum_{k=1}^{n}(-1)^{k+1}\frac{1}{k^m}\binom{n}{k}
\end{aligned}$$

Harmonic Numbers and Binomial Coefficients

From the formula $H_n^{(m)} = \sum_{k=1}^{n} (-1)^{k+1} \frac{1}{k^m} \binom{n}{k}$, one can derive some relations between harmonic numbers and binomial coefficient as follows:

$$\sum_{k=1}^{n} \frac{1}{k} H_k = H_n^{(2)} = \sum_{k=1}^{n} \frac{1}{k^2} (-1)^{k-1} \binom{n}{k}$$

$$\sum_{k=1}^{n} \frac{1}{k} (\sum_{i=1}^{k} \frac{1}{i} H_i) = \sum_{k=1}^{n} \frac{1}{k} H_k^{(2)} = H_n^{(3)} = \sum_{k=1}^{n} \frac{1}{k^3} (-1)^{k-1} \binom{n}{k}$$

$$\sum_{k=1}^{n} \frac{1}{k} H_k (H_n - H_{k-1}) = \sum_{k=1}^{n} \frac{1}{k} (\sum_{i=1}^{k} \frac{1}{i} H_i) = \sum_{k=1}^{n} \frac{1}{k^3} (-1)^{k-1} \binom{n}{k}.$$

Other formulas involving harmonic numbers and binomial coefficients can be found in [1, 2, 3] and others.

References

1. J. Choi, Finite Summation Formulas involving Binomial Coefficients, Harmonic Numbers and Generalized Harmonic Numbers, *J. Inequal. Appl.* (2013) **2013**:49
2. W. Chu and Q. Yan, Combinatorial Identities on Binomial Coefficients and Harmonic Numbers, *Util. Math.* 75 (2008) 51-66
3. M.J. Kronenburg, Some Combinatorial Identities some of which involving Harmonic Numbers, arXiv: 1103.1268v3 [math.CO] 12 Jan 2017

DECOMPOSITION OF COMPLEX VECTOR SPACE C^n INTO INVARIANT SUBSPACES

Myint Myint Maw*

Abstract

This paper study the existence of eigenvalue for every linear operator on a finite-dimensional complex vector space. In this paper, we will discuss although eigenvectors corresponding to distinct eigenvalues are linearly independent, they can not span the complex vector space. Then we give decomposition of complex vector space C^n into generalized eigenspaces and Jordan subspaces.

Keywords: Invariant subspace, Jordan chain, Generalized eigenspace, Jordan subspace

1. Eigenvalues and eigenvectors

Throughout the paper, V denotes n-dimensional complex vector space.

1.1 Definition. Let $A : V \to V$ be a linear operator. A subspace $M \subset V$ is called ***invariant*** for the linear operator A, or A-invariant, if $Ax \in M$ for every vector $x \in M$.

Trivial examples of invariant subspaces are $\{0\}, V$, Ker $A = \{x \in V \mid Ax = 0\}$ and
Im $A = \{Ax \mid x \in V\}$.

1.2 Definition. Let A: $V \to V$ be a linear operator. A number $\lambda \in C$ is called an ***eigenvalue*** of A if there exists $x \in V$ such that $x \neq 0$ and $Ax = \lambda x$. The vector x is called an ***eigenvector*** of A corresponding to λ.

1.3 Theorem. Let $A : V \to V$ be a linear operator and $\lambda \in C$. Then the following are equivalent:

(a) $\lambda$is an eigenvalue of A.

(b) $A - \lambda I$ is not injective.

*. Lecturer, Department of Mathematics, Dagon University

(c) $A - \lambda I$ is not surjective.

(d) $A - \lambda I$ is not invertible.

Proof. λ is an eigenvalue of $A \Leftrightarrow \exists v \in V$ such that $v \neq 0$ and $Av = \lambda v$.

$\Leftrightarrow (A - \lambda I)\, v = 0$

$\Leftrightarrow A - \lambda I$ is not injective.

Thus conditions (a) and (b) are equivalent.

Clearly conditions (b), (c) and (d) are equivalent.

1.4 Theorem. Every linear operator on a finite-dimensional complex vector space has an eigenvalue.

Proof. To show that A has an eigenvalvue, choose a non-zero vector $v \in V$. We consider the $n + 1$ vectors $v, Av, A^2v, \ldots, A^n v$. Since the dimension of V is n, v, $Av, A^2v, \ldots, A^n v$ are not linearly independent.

Thus there exist complex numbers $a_0, a_1, \ldots, a_n$, not all zero such that $a_0v + a_1v + \ldots + a_nA^nv = 0$.

Make the a's the coefficients of a polynomial, by the Fundamental Theorem of Linear Algebra which can be written in factored form as

$$a_0 + a_1z + \ldots + a_nz^n = c\,(z - \gamma_1) \ldots (z - \gamma_m),\ m \leq n$$

where$m$ is largest positive integer such that $a_m \neq 0$, c is a non-zero complex number, each γ_j is complex and equation holds for all complex z. We then have

$$\begin{aligned} a_0v + a_iAv + \ldots + a_nA^nv &= 0 \\ (a_0I + a_1A + \ldots + a_n A^n)v &= 0 \\ (c(A - \gamma_1I) \ldots (A - \gamma_mI))\, v &= 0. \end{aligned}$$

We know that the composition of injective mappings is injective and $v \neq 0$. Thus $A - \gamma_jI$ is not injective for at least one j. In other words, A has an eigenvalue.

1.5 Proposition. Non-zero eigenvectors corresponding to distinct eigenvalues of A are linearly independent.

Proof. Suppose that $\lambda_1, \ldots, \lambda_m$ are distinct eigenvalues of A and $v_1, \ldots, v_m$ are corresponding non-zero eigenvectors. We need to prove that $v_1, \ldots, v_m$ are linearly independent. Suppose that $a_1, \ldots, a_m$ are complex numbers such that $a_1v_1 + \ldots + a_mv_m = 0$. Apply the linear operator $(A - \lambda_2 I)(A - \lambda_3 I) \ldots (A - \lambda_m I)$ to both sides of the equation above,

$$((A - \lambda_2 I)(A - \lambda_m I) \ldots (A - \lambda_m I)(a_1v_1 + \ldots + a_mv_m) = 0.$$

Since we have $(A - \lambda_j I)\, v_j = 0,\ j = 1,2, \ldots, m$ and two polynomials in the same linear operator are commute, then we have

$$((A - \lambda_2 I)(A - \lambda_3 I) \ldots (A - \lambda_m I)(a_1v_1) = 0.$$

But $(A - \lambda_j I)\, v_1 = Av_1 - \lambda_j (Iv_1) = \lambda_1 v_1 - \lambda_j v_1 = (\lambda_1 - \lambda_j)v_1$ for $j \neq 1$.

Thus $a_1 (\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3) \ldots (\lambda_1 - \lambda_m)\, v_1 = 0$. Since λ's are distinct eigenvalues and v_1 is non-zero eigenvector, we get $a_1 = 0$. In a similar fashion, $a_j = 0$ for each j.

1.6 Definition. Suppose $A : V \to V$ and $\lambda \in \boldsymbol{C}$. The ***eigenspace*** of A corresponding to λ, denote by $E(\lambda, A)$, is defined by $E(\lambda, A) = \text{Ker}\,(A - \lambda I)$.

1.7 Theorem. Suppose V is finite-dimensional and $A: V \to V$. Suppose also that $\lambda_1, \ldots, \lambda_m$ are distinct eigenvalues of A. Then $E(\lambda_1, A) + \ldots + E(\lambda_m, A)$ is a direct sum and $\dim E(\lambda_1, A) + \ldots + \dim E(\lambda_m, A) \leq \dim V$.

Proof. We know that the null space of each linear mapping on V is a subspace of V.

Thus $E(\lambda_1, A) + \ldots + E(\lambda_m, A)$ is a subspace of V.

Take any $x \in E(\lambda_1, A) \cap E(\lambda_j, A)$ for $i \neq j$.

So $(A - \lambda_i I)x = 0$ and $(A - \lambda_j I)\, x = 0$.

$Ax = \lambda_i x$ and $Ax = \lambda_j x$ this implies that $\lambda_i x = \lambda_j x$ so $(\lambda_i - \lambda_j)x = 0$.

Since λ's are different, we get $x = 0$. Thus $E(\lambda_1, A) + \ldots + E(\lambda_m, A)$ is a direct sum of V.

Hence $\dim(E(\lambda_1, A)+\ldots+E(\lambda_m, A)) = \dim E(\lambda_1, A) +\ldots+ \dim E(\lambda_m,A) \leq \dim V$.

1.8 Remark. Non-zero eigenvectors corresponding to distinct eigenvalues of A need not span V.

1.9 Example. The linear operator $A : C^2 \to C^2$ defined by $A(w, z) = (z, 0)$.

$\forall\ (w, z) \neq (0, 0)$ and $\forall \lambda \neq 0$ in C, $\lambda\ (w, z) \neq (z, 0)$.

Thus to get $A(w, z) = \lambda\ (w, z)$, $\lambda = 0$ is forced, and so 0 is only eigenvalue of A. The set of eigenvectors corresponding 0 is $\{(w, 0) \in \mathbf{C}^2) \mid w \in \mathbf{C}\}$ it is one dimensional subspace of $\mathbf{C}^2$. Clearly $(w, 0)$ cannot span $\mathbf{C}^2$.

2. Generalized Eigenspaces

2.1 Definition. Let λ be an eigenvalue of a linear operator $A : C^n \to C^n$. A chain of vectors $x_0, x_1, \ldots, x_k$ is called ***Jordan chain*** of A corresponding to λ if $x_0 \neq 0$ and the following relation hold:

$$
\begin{aligned}
Ax_0 &= \lambda x_0 \\
Ax_1 - \lambda x_1 &= x_0 \\
Ax_2 - \lambda x_2 &= x_1 \qquad (1) \\
&\vdots \\
Ax_k - \lambda x_k &= x_{k-1}
\end{aligned}
$$

x_0 is an eigenvector of A corresponding to λ. The vectors $x_1, \ldots, x_k$ are called ***generalized eigenvectors*** of A corresponding to the eigenvalue λ and eigenvector x_0.

Equation 2.1(1) can be written $(A-\lambda I)x_0 = 0, (A - \lambda I)\, x_1 = x_0, \ldots, (A-\lambda I)x_k = x_{k-1}$.

So $(A - \lambda I)x_0 = 0$, $(A - \lambda I)^2 x_1 = 0$, $(A - \lambda I)^3 x_2 = 0$, $\ldots$, $(A - \lambda I)^{k+1} x_k = 0$. Thus we way calculate a Jordan chain into the form $(A - \lambda I)^k x_k$, $(A - \lambda I)^{k-1} x_k$, $\ldots$, $(A - \lambda I)\, x_k, x_k$.

2.2 Definition. The subspace Ker $(A - \lambda I)^p$, integer $p \geq 1$ is called the ***generalized eigenspace*** of A corresponding to eigenvalue λ of A if Ker $(A - \lambda I)^i$= Ker $(A - \lambda I)^p$ for all integer $i > p$ and is denoted by $R_\lambda(A)$. So $R_\lambda(A)$ = Ker $(A - \lambda I)^p$ is the biggest subspace in (1). Since $p \leq n$ we also have $R_\lambda(A) = \{x \in C^n \mid (A - \lambda I)^n x = 0\}$ = Ker $(A - \lambda I)^n$.

2.3 Proposition. The generalized eigenspace $R_\lambda(A)$ contains the vectors from any Jordan chain of A corresponding to λ and $R_\lambda(A)$ is A-invariant.

Proof. Let $x_0, \ldots, x_k$ be a Jordan chain of A corresponding to λ. Then

$$\begin{aligned}(A - \lambda I)^{k+1} x_k &= (A - \lambda I)^k (A - \lambda I)\, x_k \\ &= (A - \lambda I)^k x_{k-1} = (A - \lambda I)^{k-1} x_{k-2} \\ &\vdots \\ &= (A - \lambda I)\, x_0 \\ &= 0.\end{aligned}$$

Hence $x_i \in R_\lambda(A)$, $i = 0, \ldots, k$.

If $x \in$ Ker $(A - \lambda I)^n$, then $(A - \lambda I)^n x = 0$.

Thus $(A - \lambda I)^n (Ax) = A((A - \lambda I)^n x) = A0 = 0$.

Hence $R_\lambda(A)$ = Ker $(A - \lambda I)^n$ is A-invariant.

2.4 Lemma. For any eigenvalue λ of A, then (the restriction linear operator of A on $R_\lambda(A)$), $A|_{R_\lambda(A)}$ has only one eigenvalue λ.

Proof. Let λ' be eigenvalue of $A|_{R_\lambda(A)}$.

Then there exists nonzero eigenvector $x \in R_\lambda(A)$ such that $Ax = \lambda' x$. Then

$(A - \lambda I)\, x = \lambda' x - \lambda x = (\lambda' - \lambda)\, x$

$(A - \lambda I)^2 x = (A - \lambda I)\,(\lambda' - \lambda)x = (\lambda' - \lambda)(\lambda' - \lambda)x = (\lambda' - \lambda)^2 x$ and so on, thus we have $(A - \lambda I)^k x = (\lambda' - \lambda)^k x$ for each positive integer k. Since x is

generalized eigenvector of A corresponding λ, for some l, then $(\lambda' - \lambda)^l = 0$, thus we have $\lambda' = \lambda$.

2.5 Lemma. If $A : C^n \to C^n$ be a linear operator, then non zero generalized eigenvectors corresponding to distinct eigenvalues of A are linearly independent.

Proof. Suppose $\lambda_1, \ldots, \lambda_m$ are distinct eigenvalues of A and $v_1, \ldots, v_m$ are corresponding non zero generalized eigenvectors. Suppose

(1) $a_1 v_1 + \ldots + a_m v_m = 0$ for some scalars $a_1, \ldots, a_m$.

Let k be the largest non negative integer such that $(A - \lambda_1 I)^k v_1 \neq 0$ and $(A - \lambda_1 I)^k v_1 = w$. Thus $(A - \lambda_1 I)\, w = (A - \lambda_1 I)^{k+1} v_1 = 0$ and hence $Aw = \lambda_1 w$. Thus $(A - \lambda I)\, w = \lambda_1 w - \lambda w = (\lambda_1 - \lambda) w$, $\forall \lambda \in C$. So $(A - \lambda I)^n w = (\lambda_1 - \lambda)^n w$, $\forall \lambda \in C$, where $n = \dim C^n$. Apply the linear operator

$$(A - \lambda_1 I)^k (A - \lambda_2 I)^n \ldots (A - \lambda_m I)^n \text{ to (1)}$$

$$(A - \lambda_1 I)^k (A - \lambda_2 I)^n \ldots (A - \lambda_m I)^n (a_1 v_1 + \ldots + a_m v_m) = 0$$

$$a_1 (A - \lambda_1 I)^k (A - \lambda_2 I)^n \ldots (A - \lambda_m I)^n v_1 = 0$$

$$a_1 (A - \lambda_2 I)^n \ldots (A - \lambda_m I)^n w = 0$$

$$a_1 (\lambda_1 - \lambda_2)^n \ldots (\lambda_1 - \lambda_m)^n w = 0.$$

This implies that $a_1 = 0$. In a similar fashion $a_j = 0$ for each j. Thus $v_1, \ldots, v_m$ are linearly independent.

2.6 Lemma. Given a linear operator $A : C^n \to C^n$ with an eigenvalue λ, let q be a positive integer for which Ker $(A - \lambda I)^q = R_\lambda(R)$. Then the subspace Ker $(A - \lambda I)^q$ and Im $(A - \lambda I)^q$ are direct complements to each other in C^n.

Proof. Since dim Ker $(A - \lambda I)^q$ + dim Im $(A - \lambda I)^q = n$, we have only to check that Ker $(A - \lambda I)^q \cap$ Im $(A - \lambda I)^q = \{0\}$.

For a contradiction, assume that $x \in$ Ker $(A - \lambda I)^q \cap$ Im $(A - \lambda I)^q$, $x \neq 0$.

Then $x = (A - \lambda I)^q y$, for some y and $(A - \lambda I)^r x = 0$ and $(A - \lambda I)^{r-1} x \neq 0$ for some integer $r \geq 1$. Thus $(A - \lambda I)^{q+r} y = 0$ and $(A - \lambda I)^{q+r-1} y \neq 0$. So Ker

$(A - \lambda I)^{q+r} \neq$ Ker $(A - \lambda I)^{q+r-1}$. This contradicts to definition of generalized eigenspace.

2.7 Theorem. Let $\lambda_1, \ldots, \lambda_r$ be all the different eigenvalues of a linear operator $A : C^n \to C^n$. Then C^n decomposes into the direct sum $C^n = R_{\lambda_1}(A) + \ldots + R_{\lambda_r}(A)$.

Proof. For $n = 1$. let λ be an eigenvalue of A, then there exists $v \neq 0$ in C^n such that $Av = \lambda v$. Since $\{v\}$ is a basic of C^n, for each $x \in C^n$ $(A - \lambda I)x = (A - \lambda I)\mu v$ for some $\mu \in C$. So we have $(A - \lambda I)x = \mu\lambda v - \lambda\mu v = 0$. Then $x \in R_\lambda(A)$. Thus $C^n = R_\lambda(A)$.

Let $n > 1$. Assume that the result holds for dimensions $k = 1, 2, \ldots, n - 1$.

Consider the eigenvalue λ_1.

$C^n =$ Ker $(A - \lambda_1 I)^n +$ Im $(A - \lambda_1 I)^n = R_{\lambda_1}(A) + U$. We know that Im $(A - \lambda_1 I)^n = U$ is A-invariant. Since $R_{\lambda_1}(A) \neq 0$, we have dim $U < n$. By Proposition 2.3, there does not exist generalized eigenvectors of $A|_U$ corresponding to the eigenvalue λ_1. Thus each eigenvalue of $A|_U$ corresponding to the eigenvalue λ_1. Thus each eigenvalue of $A|_U$ is in $\{\lambda_2, \ldots, \lambda_r\}$. By induction hypothesis $U = R_{\lambda_2}(A|_U) + \cdots + R_{\lambda_r}(A|_U)$. Thus $C^n = R_{\lambda_1}(A) + R_{\lambda_2}(A|_U) + \cdots + R_{\lambda_r}(A|_U)$. So we show that $R_{\lambda_k}(A) = R_{\lambda_k}(A|_U)$ for $k = 2, \ldots, m$. Take a fixed integer $k \in \{2, \ldots, m\}$ and clearly $R_{\lambda_k}(A|_U) \subseteq R_{\lambda_k}(A)$. Assume $R_{\lambda_k}(A|_U) \neq R_{\lambda_k}(A)$. Then there exists $v \in R_{\lambda_k}(A)$ but $v \notin R_{\lambda_k}(A|_U)$. So we get $v \in R_{\lambda_j}(A|_U)$ for some $j \neq k$ and hence $v \in R_{\lambda_j}(A)$. Thus $v \in R_{\lambda_k}(A) \cap R_{\lambda_j}(A)$. This contradicts to lemma 2.5. So $R_{\lambda_k}(A) = R_{\lambda_k}(A|_U)$. Thus $C^n = R_{\lambda_1}(A) + \cdots + R_{\lambda_r}(A)$.

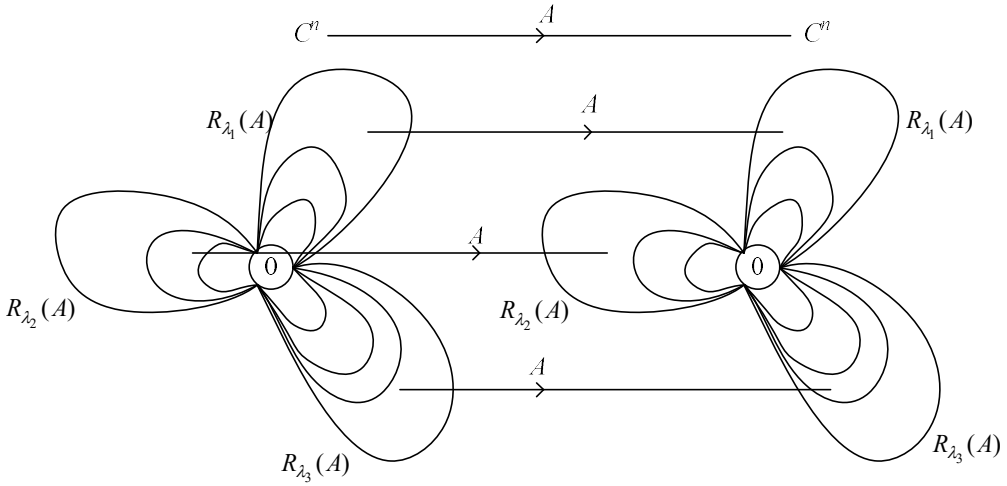


Figure 1

3. Jordan Subspaces

3.1 Definition. An A-invariant subspace M is called a ***Jordan subspace*** corresponding the eigenvalue λ_0 of A if M is spanned by the vectors of some Jordan chain of A corresponding to λ_0.

3.2 Proposition. Let $A : C^n \to C^n$ be a linear operator. Let $x_0, x_1, \ldots, x_k$ be a Jordan chain of a linear operator A corresponding to λ_0. Then the subspace $M = \text{Span}\{x_0, \ldots, x_k\}$ is A-variant.

Proof. We have $Ax_0 = \lambda_0 x_0 \in M$ where λ_0 is the eigenvalue of A and for $i = 1, \ldots, k$, $Ax_i = \lambda_0 x_i + x_{i-1} \in M$. Hence M is A-invariant.

3.3 Theorem. Let $A : C^n \to C^n$ be a linear operator. Then there exists a direct sum decomposition

$$C^n = M_1 + \ldots + M_p \tag{1}$$

where M_i is a Jordan subspace of A corresponding to an eigenvalue λ_i ($\lambda_1, \ldots, \lambda_p$ are not necessarily different).

Proof. Assume A has only one eigenvalue λ_0, (possibly with there are more one eigenvalue, all equal to λ_0).

Let $Y_j = \text{Ker}\,(A - \lambda_0 I)^j$, $j = 1, 2, \ldots, m$, where m is chosen $Y_m = R_{\lambda_0}(A)$ and $Y_{m-1} \neq R_{\lambda_0}(A)$. So $Y_1 \subset Y_2 \subset \ldots \subset Y_m$. Let $x_m^{(1)}, \ldots, x_m^{(t_m)}$ is a basis of Y_m modulo Y_{m-1}. So $x_m^{(1)}, \ldots, x_m^{(t_m)}$ are linearly independent in set Y_m such that

$$Y_{m-1} + \text{Span}\,\{x_m^{(1)}, \ldots, x_m^{(t_m)}\} = Y_m \quad \text{(the sum is here direct)} \tag{2}$$

Claim that the mt_m vectors $(A-\lambda_0 I)^k x_m^{(1)}, \ldots, (A-\lambda_0 I)^k x_m^{(t_m)}$, $k = 0, \ldots, m-1$ are linearly independent. Let

$$\sum_{k=0}^{m-1} \sum_{i=1}^{t_m} \alpha_{ik} (A-\lambda_0 I)^k x_m^{(i)} = 0, \qquad \alpha_{ik} \in C\,. \tag{3}$$

Apply $(A - \lambda_0 I)^{m-1}$ and use the property $(A-\lambda_0 I)^m x_m^{(i)} = 0$, for $i = 1, \ldots, t_m$.

Thus $(A-\lambda_0 I)^{m-1} \left\{ \sum_{i=1}^{t_m} \alpha_{i0} x_m^{(i)} \right\} = 0$. So $\sum_{i=1}^{t_m} \alpha_{i0} x_m^{(i)} \in Y_{m-1}$.

By 3.3(2), $\sum_{i=1}^{t_m} \alpha_{i0} x_m^{(i)} \in Y_{m-1} \cap \text{Span}\,\{x_m^{(1)}, \ldots, x_m^{t_m}\}$ and so $\alpha_{10} = \cdots = \alpha_{t_m 0} = 0$.

Apply $(A - \lambda_0 I)^{m-2}$ to 3.3(3) we show similarly that $\alpha_{11} = \cdots = \alpha_{t_m 1} = 0$ and so on.

We put

$$M_1 = \text{Span}\,\{(A-\lambda_0 I)^k x_m^{(1)},\ k = 0, \ldots, m-1\}$$

$$M_2 = \text{Span}\,\{(A-\lambda_0 I)^k x_m^{(2)},\ k = 0, \ldots, m-1\}$$

$$\vdots$$

$$M_{t_m} = \text{Span}\,\{(A-\lambda_0 I)^k x_m^{(t_m)},\ k = 0, \ldots, m-1\}.$$

Since $M_i \cap M_j = \{0\}$ for $i \neq j$, then the sum $M_1 + M_2 + \ldots + M_{t_m}$ is direct. Now consider the linear independent vectors $x_{m-1}^{(i)} = (A - \lambda_0 I)x_m^{(i)}$, $i = 1, \ldots, t_m$. Claim that

(4) $$Y_{m-2} \cap \text{ Span } \{x_{m-1}^{(1)}, x_{m-1}^{(2)}, \ldots, x_{m-1}^{(t_m)}\} = \{0\}.$$

Let $\sum_{i=1}^{t_m} \alpha_i x_{m-1}^{(i)} \in Y_{m-2}$, $\alpha_i \in C$. Apply $(A - \lambda_0 I)^{m-2}$ to the left-hand side, we get

$$(A-\lambda_0 I)^{m-2} \sum_{i=1}^{t_m} \alpha_i (A-\lambda_0 I)x_m^{(i)} = 0, \quad \text{So} \quad (A-\lambda_0 I)^{m-1} \sum_{i=1}^{t_m} \alpha_i x_m^{(i)} = 0, \quad \text{which}$$

implies $\alpha_1 = \cdots = \alpha_{t_m} = 0$. So the equation 3.3(4) follows. Assume that $Y_{m-2} + \text{Span } \{x_{m-1}^{(1)}, \ldots, x_{m-1}^{(t_m)}\} \neq Y_{m-1}$. Then there exist vectors $x_{m-1}^{(t_m+1)}, \ldots, x_{m-1}^{(t_m+t_{m-1})} \in Y_{m-1}$ such that $\{x_{m-1}^{(i)}\}_{i=1}^{t_m+t_{m-1}}$ is linearly independent and

(5) $$Y_{m-2} + \text{Span } \{x_{m-1}^{(1)}, \ldots, x_{m-1}^{(t_m+t_{m-1})}\} = Y_{m-1}.$$

Applying previous argument to 3.3(5) as with 3.3(2), we fine that the vectors $(A-\lambda_0 I)^k x_{m-1}^{(1)}, \ldots, (A-\lambda_0 I)^k x_{m-1}^{(t_m+t_{m-1})}, k = 0, \ldots, m-2$ are linearly independent.

Now let $$M_{t_m+1} = \text{ Span } \{(A-\lambda_0 I)^k x_{m-1}^{(t_m+1)}, k = 0, \ldots, m-2\}$$

$$\vdots$$

$$M_{t_m+t_{m-1}} = \text{ Span } \{(A-\lambda_0 I)^k x_{m-1}^{(t_m+t_{m-1})}, k = 0, \ldots, m-2\}.$$

If $Y_{m-2} + \text{Span}\{x_{m-1}^{(1)}, \ldots, x_{m-1}^{(t_m)}\} = Y_{m-1}$, then $t_{m-1} = 0$.

At the next step put $x_{m-2}^{(i)} = (A - \lambda_0 I)x_{m-1}^{(i)}$, $i = 1, \ldots, t_m + t_{m-1}$ and show similarly that $Y_{m-3} \cap \text{Span}\{x_{m-2}^{(i)}, i = 1, \ldots, t_m + t_{m-1}\} = \{0\}$.

Assume that that $Y_{m-3} \cap \text{Span}\{x_{m-2}^{(i)}, i = 1,\ldots,t_m + t_{m-1}\} \neq Y_{m-2}$, then there exist vectors $x_{m-2}^{(i)} \in Y_{m-2}$, $i = t_m + t_{m-1} + 1,\ldots,t_m + t_{m-1} + t_{m-2}$ such that $x_{m-2}^{(i)}$, $i = 1,\ldots,t_m + t_{m-1} + t_{m-2}$ are linearly independent and $Y_{m-3} + \text{Span}\ \{x_{m-2}^{(i)}, i = 1,\ldots,t_m + t_{m-1} + t_{m-2}\} = Y_{m-2}$.

We continue this process of construction of M_i, i = 1, …,p where $p = t_m + t_{m-1} + \ldots + t_1$.

The construction shows that each M_i is Jordan subspace of A and M_1 + … + M_p is a direct sum. Also $M_1 + \ldots + M_p = R_{\lambda_0}(A) = C^n$.

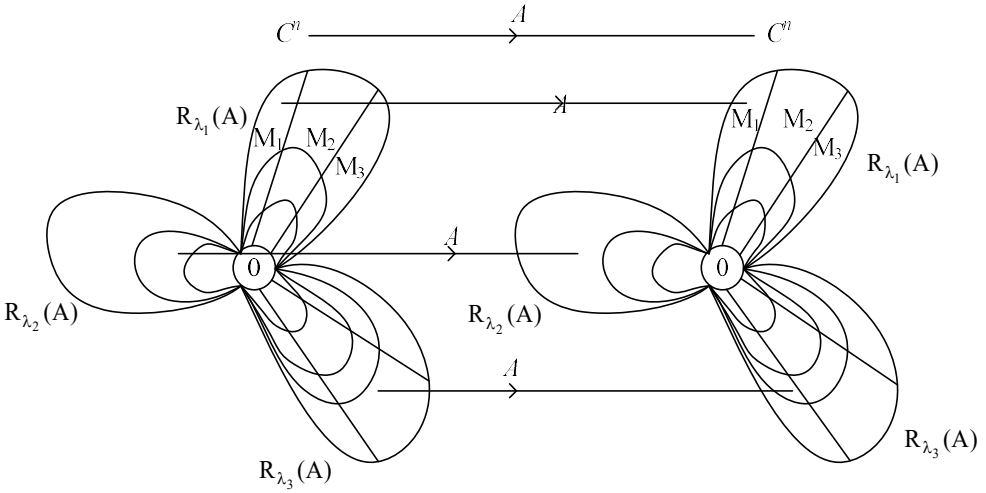


Figure. 2

3.4 Example. Let us consider the matrix

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$|A - \lambda I| = (2 - \lambda)^5$

λ= 2, 2, 2, 2, 2 are eigenvalues of A.

$$A-2I=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&1\\0&0&0&0&0\end{bmatrix},(A-2I)^2=\begin{bmatrix}0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},(A-2I)^2=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}$$

$Y_1 = Ker(A-2I) = \{\lambda_1 e_1 + \lambda_2 e_4 \mid \lambda_1, \lambda_2$ are scalars$\}$

$Y_2 = Ker(A-2I)^2 = \{\mu_1 e_1 + \mu_2 e_2 + \mu_3 e_4 + \mu_4 e_5 \mid \mu_1, \mu_2, \mu_3, \mu_4$ are scalars$\}$

$Y_3 = Ker(A-2I)^3 = \{\eta_1 e_1 + \eta_2 e_2 + \eta_3 e_3 + \eta_4 e_4 + \eta_5 e_5 \mid \eta_1, \eta_2, \eta_3 + \eta_4 + \eta_5$ are scalars$\}$

$$Y_1 \subset Y_2 \subset Y_3 = R_2(A) = C^5$$

e_3 is a basis of Y_3 modulo Y_2 such that

$$Y_2 + \text{span } \{e_3\} = Y_3$$

Jordan subspace $M_1 = \text{Span}\{(A-2I)^2 e_3, (A-2I)e_3, e_3\} = Span\{e_1, e_2, e_3\}$

$$(A-2I)e_3 = e_2 \in Y_2$$

$$Y_1 + \text{Span}\{(A-2I)e_3\} \neq Y_2$$

$\exists e_5 \in Y_2$ such that $\{e_2 , e_5\}$ is linearly independent set.

Jordan subspace M_2= Span $\{(A-2I)e_5, e_5\} = Span\{e_4, e_5\}$

$$M_1 + M_2 = R_2(A) = C^5 = Y_3$$

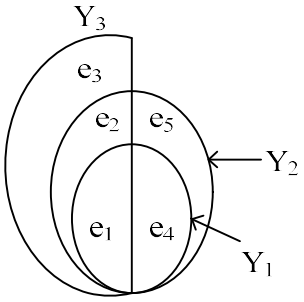


Figure. 3

References

1. S. Axler, Down with Determinants!, *American Mathematical Monthly* **102**(1995) 139-154.
2. S. Axler, *Linear Algebra Done Right*, Third Edition, Springer International Publishing, 2015.
3. I. Gohberg, P.Lancaster & L. Rodman, *Invariant Subspaces of Matrices with Application*, John Wiley & Sons, New York, 2006.
4. P. R. Halmos, *Finite-Dimensional Vector Spaces*, Second Edition, Springer-Verlag, New York, 1987.

EIGENVALUES OF SOME COMPOSITE GRAPHS

Zaw Win[1] and Aung Kyaw[2]

Abstract

K_n is a complete graph with n vertices. $K_n^{(m)}$ is a graph containing m copies of K_n with each vertex of a K_n is only adjacent to a vertex of each of the other K_n.

We will show that the adjacency matrix of $K_n^{(m)}$ has

(i) $(n-1)(m-1)$ eigenvalues of -2

(ii) $m-1$ eigenvalues of $n-2$

(iii) $n-1$ eigenvalues of $m-2$

(iv) an eigenvalue of $n+m-2$.

Composite Graph $K_n^{(m)}$ and Its Adjacency Matrix

K_n is a complete graph with n vertices. $K_n^{(m)}$ is a graph containing m copies of K_n with each vertex of a K_n is only adjacent to a vertex of each of the other K_n. (See figure 1 for $K_3^{(4)}$)

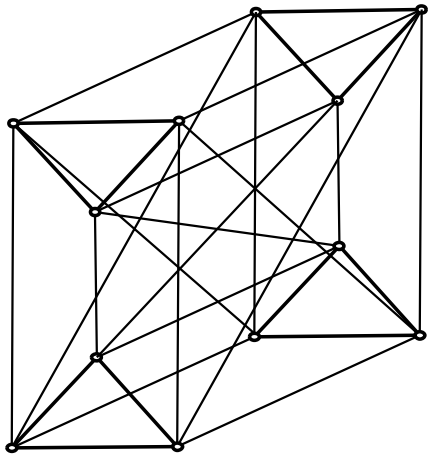


Figure 1. $K_3^{(4)}$

[1.] Professor & Head (Retd.), Department of Mathematics, University of Yangon
[2.] Professor, Department of Mathematics, University of Yangon

Basic definitions and notations on graphs and their eigenvalues can be found in [1, 2, 3] and others.

The adjacency matrix of $K_n^{(m)}$ is like as follow:

$$\begin{bmatrix} K_n & I_n & I_n & \cdots & I_n \\ I_n & K_n & I_n & \cdots & I_n \\ I_n & I_n & K_n & \cdots & I_n \\ I_n & I_n & I_n & \ddots & I_n \\ I_n & I_n & I_n & I_n & K_n \end{bmatrix}$$

For example, adjacency matrix of $K_3^{(4)}$ is

$$\left[\begin{array}{ccc:ccc:ccc:ccc} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Since adjacency matrix $\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \ddots & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ of K_n is like as $\begin{bmatrix} K_1 & I_1 & I_1 & \cdots & I_1 \\ I_1 & K_1 & I_1 & \cdots & I_1 \\ I_1 & I_1 & K_1 & \cdots & I_1 \\ I_1 & I_1 & I_1 & \ddots & I_1 \\ I_1 & I_1 & I_1 & \cdots & K_1 \end{bmatrix}$,

$K_n^{(m)}$ can be seen as a generalization of complete graphs.

Eigenvalues and Eigenvectors of Adjacency Matrix of $K_n^{(m)}$

The adjacency matrix of $K_n^{(m)}$ has

(i) $(n-1)(m-1)$ eigenvalues of -2

(ii) $m-1$ eigenvalues of $n-2$

(iii) $n-1$ eigenvalues of $m-2$

(iv) an eigenvalue of $n+m-2$.

By using each of the eigenvectors shown in figure 2, one can check that there are $(n-1)(m-1)$ eigenvalues of -2.

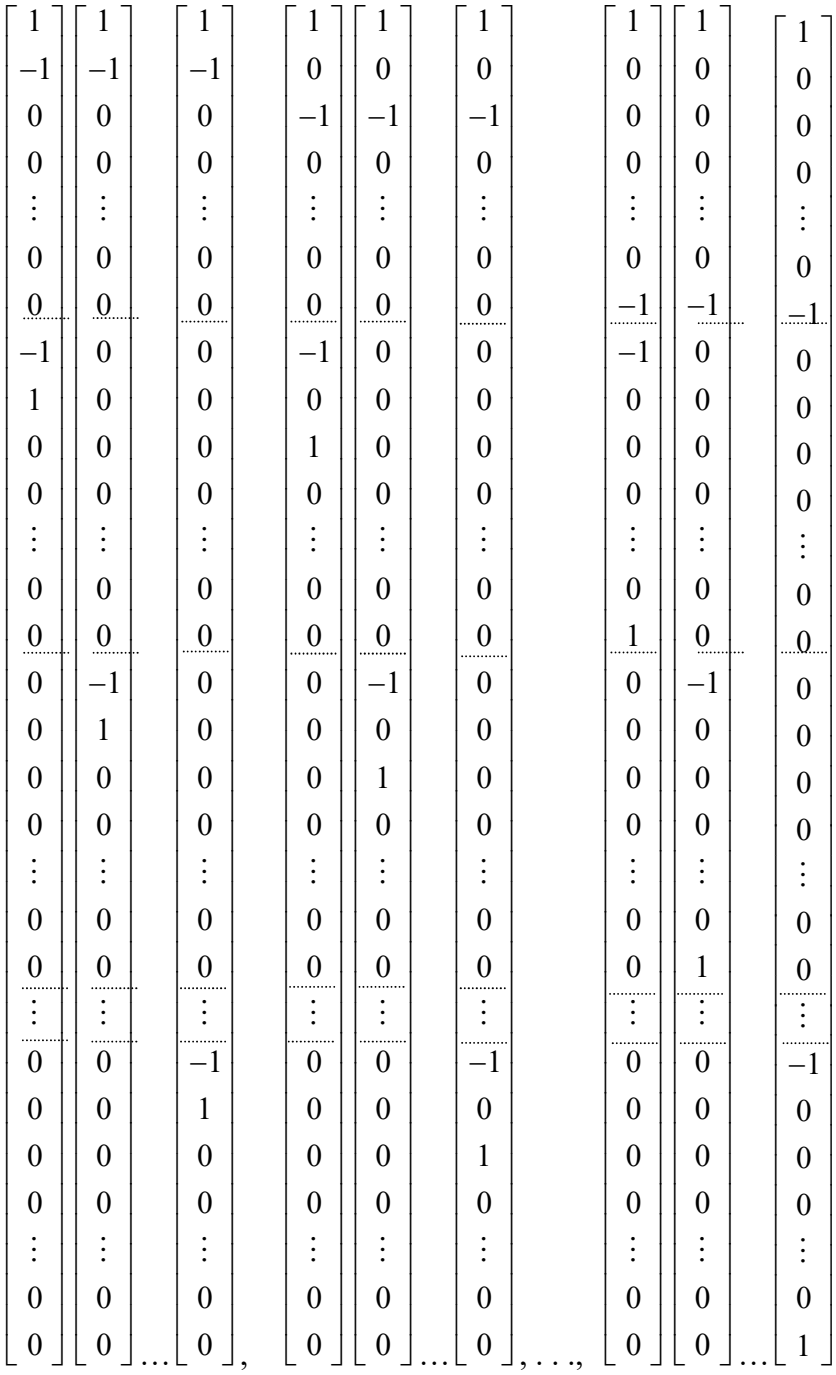


Figure 2. $(n-1)(m-1)$ eigenvectors of eigenvalue -2

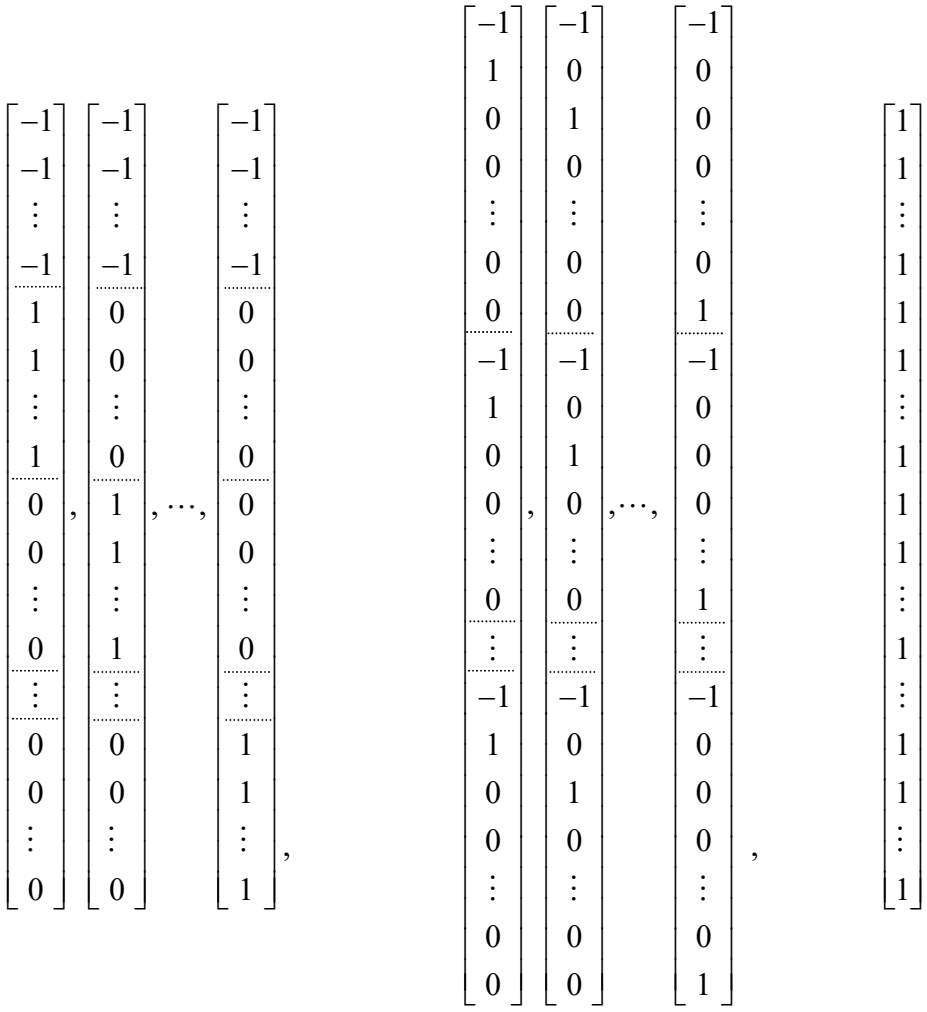


Figure 3. (a) $m-1$ eigenvectors of eigenvalue $n-2$;
(b) $n-1$ eigenvectors of eigenvalue $m-2$; (c) an eigenvector of eigenvalue $n+m-2$.

According to eigenvectors shown in figure 3, there are m–1 eigenvalues of $n-2$, $n-1$ eigenvalues of $m-2$ and an eigenvalue of $n+m-2$.

Acknowledgements

The Second Author was supported by the Asia Research Center under project code ARCYU/002/2014-15/MATHEMATICS.

References

1. R.B. Bapat, *Graphs and Matrices*, 2nd Edition, Springer, 2014
2. S.K. Butler, *Eigenvalues and Structures of Graphs*, PhD Dissertation, University of California, San Diego, 2008
3. C. Godsil and G.F. Royle, *Algebric Graph Theory*, Springer, 2001

USING GRAPH DATABASE FOR EFFECTIVE VISUALIZATION IN LEARNING BASIC BUDDHIST VOCABULARY*

Ohnmar Win[1]

Abstract

Graph databases have a long academic tradition. At the heart of any graph database lies an efficient representation of entities and relationships between them. All graph database models have, as their formal foundation, variations on the basic mathematical definition of a graph, for example, directed or undirected graphs, labeled or unlabeled edges and nodes, hypergraphs, and hypernodes. More recently, semantic relations have become a major theme of interest of Computational Linguistics. Semantic relations among words have captured the interest of various brands of philosophers, cognitive psychologists, linguists, early childhood and second language educators, computer scientists, literary theorists, cognitive neuroscientists, psychoanalysts - investigators from just about any field whose interests involve words, meaning or the mind. The Pāḷi Canon is the complete scripture collection of the Theravāda school. Buddhist monks and scholars studied the Pāḷi language mainly to gain access to the Buddhist Canon and many religious works were written using the Pāḷi language. The objective of this study is to support for new Buddhist vocabulary learner to alternative view by using graph database, Neo4j.

Keywords: graph database, semantic relations, Neo4j, Pāḷi, Buddhist Vocabulary

Introduction

Semantics is the study of the relationship between the linguistics forms and entities in the world, that is, how words literally connect to things (meaning). It is a major branch of linguistics devoted to the study of meaning in language. In many research fields such as linguistics, cognitive science, psychology, artificial intelligence, biomedicine and information retrieval, computing semantic similarity/relatedness between concepts or words is considered as an important issue. More recently, semantic relations have become a major theme of interest of Computational Linguistics, as they present a convenient and natural way to organize huge amounts of lexical data in ontologies, Word Nets and other machine-readable lexical resources.

1. Lecturer, Department of Computer Studies, Yadanabon University

*. Best Paper Award Winning Paper in Computer Studies (2017)

Semantic relations may reflect relations in language including relations between objects and their symbols. Semantic relations can refer to relations *between concepts in the mind* (called conceptual relations), or relations *between words* (lexical relations) or text segments. Different domains develop continuously new kinds of semantic relations. Some kind of semantic relationships that exist in words of natural language have always been a challenge in the Fields of Natural Language Processing (NLP) and Information Retrieval (IR). When a word level semantic relation requires exploration, there are many potential types of relations that can be considered: synonym, antonym, homonym, polysemy, hyponym, meronym, etc. Semantic relations are fixed manually in various linguistic resources, such as thesauri, ontologies, and synonym dictionaries.

The relationships between words can be summarized briefly as follows:

Synonym : The notion that more than one linguistic form can be said to have the same conceptual or propositional meaning.

e.g., Nibbāna and Mokkha

Antonym : The notion of semantic oppositeness.

e.g., Amitta and mitta

Hyponym : Refers to a relationship existing between specific and general lexical items: the meaning of the specific item is included in, and by, the meaning of the more general item.

e.g., sunakha is a hyponym of tiricchāna.

Meronym : Refers to a part-whole relation.

e.g., Rukkha and Phala

The limitations of traditional databases, in particular the relational model, to cover the requirements of current application domains, has lead the development of new technologies called Graph Databases, which are oriented to store graph-like data. Recently the area is gaining attention because in trendy projects where a database is needed (for example chemistry, biology, Web Mining and semantic Web), the importance of the information relies on the relations more or equal than on the entities. Moreover, the continued

emergence and increase of massive and complex graph-like data makes a graph database a crucial requirement. This renascence is showed by the availability of several graph databases systems.

With the needs to manage large and sparse datasets, with many kinds of relationships between them, new kinds of Database have been developed to supply it with a performance and capability better than the traditional databases technologies and queries languages. Many of these new kinds of Databases using graph structures like the main engine to allow to user to insert, update, query, delete and apply analysis techniques based in graphs in the networks of graphs.

Graph Database is a database system where the associations between objects or entities are similarly as important as the objects themselves. In a graph database, data are represented by nodes, edges and properties. Nodes are represented as objects and edges manifest the relationship between nodes. There are several implementations of graphical database. Both nodes and edges can have properties that illustrate their particular characteristics.

Graph databases are especially suited for highly connected data. Today, general-purpose graph databases are a reality, allowing mainstream users to experience the benefits of connected data without having to invest in building their own graph infrastructure. Today, there are many graph databases such as Allegro Graph, DEX/Sparkee, Hypergraph DB, Infinite Graph, Neo4J, Orient DB, Info Grid, Vertex DB, Flock DB, Graph DB etc.

An Overview of Neo4j Graph Database

Neo4j is the world's leading graph database. Neo4j is a high performance graph store with all the features expected of a mature and robust database, like a friendly query language and ACID transactions.

Neo4j is a graph database, which means that it does not use tables and rows to represent data logically; instead, it uses nodes and relationships. Both nodes and relationships can have a number of properties. While relationships must have one direction and one type, nodes can have a number of labels. The programmer works with a flexible network structure of nodes and relationships rather than static tables. For many applications, Neo4j offers orders of magnitude performance benefits compared to relational databases.

Neo4j is based on a network oriented model where relations are first class objects.

The most popular variant of graph model is the property graph. Property graphs are attributed, labeled, directed multi-graphs. The property graph balances simplicity and expressiveness. Property graphs sacrifice some graph purity for pragmatism by grouping properties into nodes, thereby making them easier to work with. The main abstractions in a property graph are nodes, relationships and properties. Neo4j uses Cypher Query Languages for property graphs. A Property graph has the following characteristics:

- It contains nodes and relationships
- Nodes contain properties
- Relationships are named, directed and always have a start and end node
- Relationships can also contain properties

Most people find the property graph model intuitive and easy to understand.

Neo4j has many features. The main feature is that neo4j not depend heavily on index because it supplies a natural adjacency by the graph. It is easy to write queries about relationships with many types of deep.

The Cypher Query Language in Neo4j

Cypher is a declarative graph query language that allows for expressive and efficient querying and updating of the graph store. Cypher is designed to be a humane query language, suitable for both developers and operations professionals who want to make ad hoc queries on the database. Cypher is a database expressive and compact query language. It is primarily used in Neo4j, although it can also be used to programmatically describe graphs in a precise manner due to its close affinity to graphs. It is easy to learn and understand since it follows the way humans intuitively describe graphs using diagrams. Cypher is a relatively simple but still very powerful language. Very complicated database queries can easily be expressed through Cypher. Like most query languages, Cypher is composed of clauses. A reasonably simple query is made up of START, MATCH and RETURN clauses.

The some clauses of Cypher are:

- START – specifies one or more starting points – nodes or relationships – in a graph, which are obtained via index lookup (starting points are rarely accessed via IDs).
- MATCH – it makes use of the relationships
- RETURN – returns nodes and relationships that match the criteria
- WHERE – acts as a filter pattern for matching results
- CREATE or CREATE UNIQUE – creates (unique) nodes and relationships
- DELETE – removes nodes, relationships or properties
- SET – sets property values
- UNION – merges results from two or more queries
- WITH – chains subsequent query results and pipelines results

The Operations of Neo4j Graph Database

Neo4j has CRUD operations. They are Create, Read, Update, and Delete.

CREATE Operation

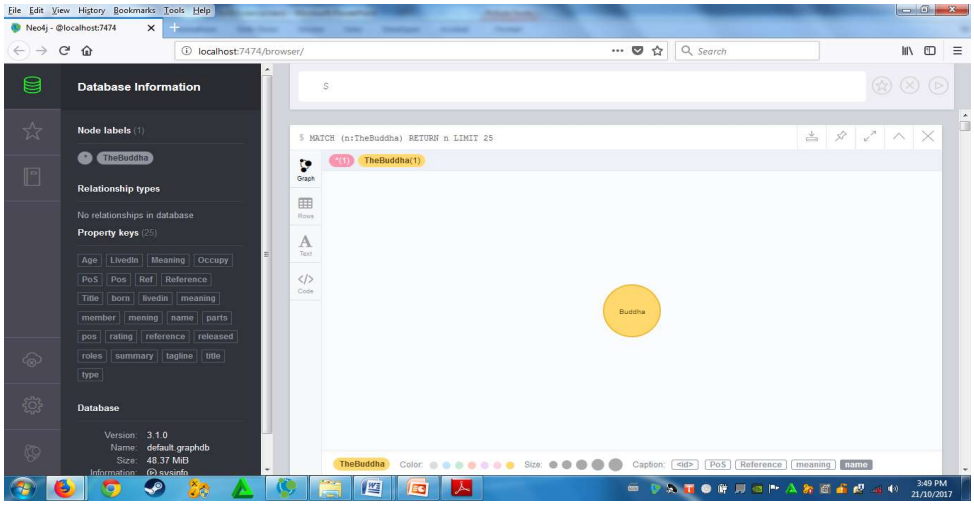
Create operation is used to create nodes and relationships.

e.g., Creating a node

```
create(a:TheBuddha{name: 'Buddha', meaning: 'Supreme Man',
PoS: 'Noun', Reference: 'Pāḷi Canon'})
```

Create clauses can create nodes and relationships.***( ) parenthesis*** is to indicate a node. In ***a:TheBuddha***, '***a***' is variable and***The Buddha***is label for the new node. ***{}*** bracket can be used to add properties to the node.

Result in console:



-Creating multiple nodes

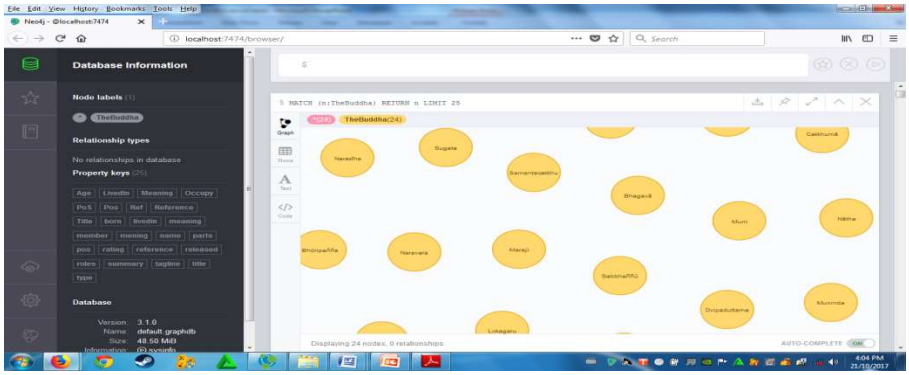
```
create (a:The Enlightened One {name: "Buddha", meaning: "Supreme
Man", PoS:"Noun", Reference: "PāḷiCanon"}), (b:The Enlightened
One{name:" Dasabala", meaning:"Ten powers of Buddha", PoS: "Noun",
Reference: "Pāḷi Canon"}), (c:The EnlightenedOne {name:"Satthā",
meaning:"A Supreme teacher", PoS:"Noun", Reference:
"PājikakaṇḍaPāḷi"}), etc.,
```

In the following table, there are some of the epithets of the Buddha and its properties.

Table 1. The epithets of the Buddha and its properties.

Name	Meaning	PoS	Reference
Dasabala	Ten powers of Buddha	Noun	Pāḷi Canon
Satthā	A Supreme teacher	Noun	PārājikakaṇḍaPāḷi
Sabbaññū	All-Knowing	Noun	Theragāthāand MajjhimapaṇṇāsaPāḷi
Dvipaduttama	The best of Men	Noun	Buddhavaṁsa
Muninda	The chief of monks	Noun	ApadānaPāḷi
Bhagavā	The Blessed One	Noun	PārājikakaṇḍaAṭṭhakathā
Nātha	Protector	Noun	three piṭakas
Cakkhumā	Having eyes	Noun	Pāḷi Canon
Muni	Monk	Noun	Pāḷi Canon
Lokanātha	TheRefuge of the human beings	Noun	Suttapiṭaka
Anadhivara	There is no one who is superior to oneself	Noun	Buddhavṁsa
Mahesi	The Great Sage	Noun	Pāḷi Canon
Vināyaka	One who admonishes the living beings	Noun	Sutta and VinayaPiṭakas
Samantacakkhu	All-Seeing	Noun	Pāḷi Canon
Sugata	Meritorious act	Noun	Five Nikāyas
Bhūripañña	abundant knowledge as the earth	Noun	Majjhimapaṇṇāsa
Maraji	Supreme Man	Noun	Pāḷi Canon
Narasīha	The Noble Man	Noun	TheragāthāPāḷi
Naravara	The Noble Man	Noun	TheragāthāPāḷi
Dhammarājā	The King of righteousness	Noun	Theragāthā
Mahāmuni	The Great Sage	Noun	Suttaand VinayaPiṭakas
Devadeva	The God of gods	Noun	Theragāthā and ApadānaPāḷi
Lokagaru	The One who is a teacher deserving the special veneration of human beings	Noun	SakulātherīApadāna

Result with graph view for creation of multiple nodes;



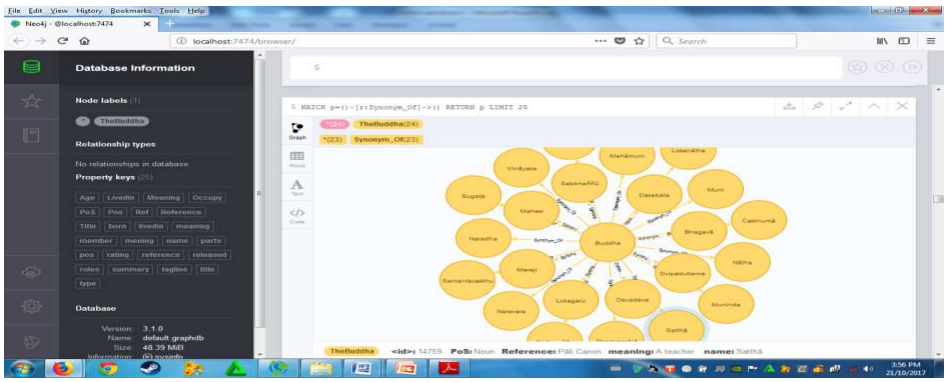
-Creation of relationships

These are coding for relationships which are created after the creation of nodes' codes under the following:

```
create
 (a)-[:synonym_of]->(b),(a)-[:synonym_of]->(c),(a)-[:synonym_of]->(d),
 (a)-[:synonym_of]->(e),(a)-[:synonym_of]->(f),(a)-[:synonym_of]->(g)
```

In Relationships, Cypher uses a pair of dashes (--) to represent undirected relationship. Directed relationships have an arrow head at one end (eg.,<-- , -->). Bracketed expression may include types of relationships, properties, and attributes.

Result with graph view for creation of relationships between nodes;

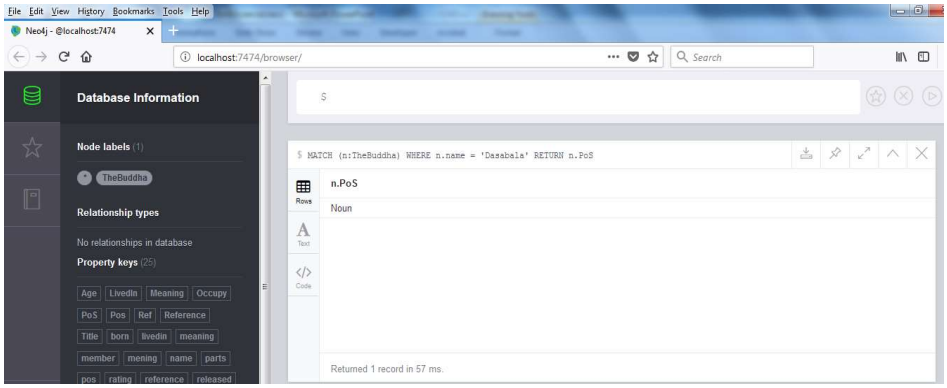


Read Operation

Reading a node named ‘Dasabala’ and return the PoS.

e.g.,
```
MATCH (n:TheEnlightenedOne) WHERE n.name =‘Dasabala' RETURN
n.Pos
```

Result in console:

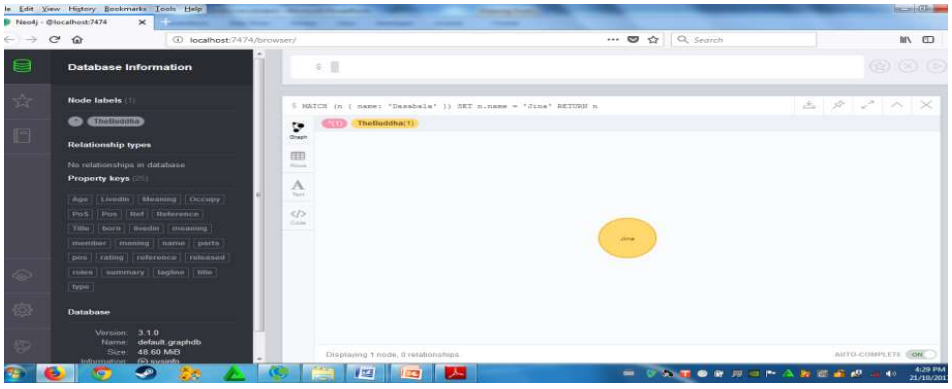


Update Operation

-Updating a node named ‘Dasabala’ with 'Jina':

e.g.,
```
MATCH (n {name: ‘Dasabala'}) SET n.title = ‘Jina' RETURN n
```

Result in console:

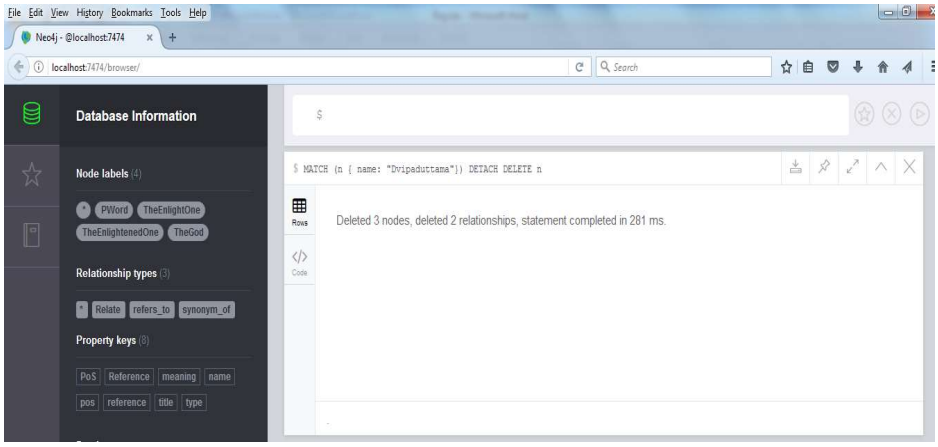


Delete Operation

- Deletinga node named with "Dvipaduttama"and all its relationships

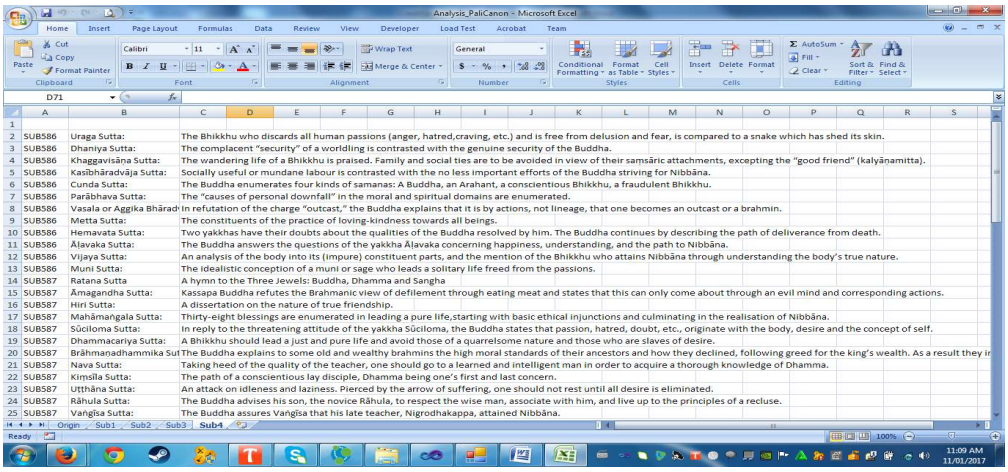
e.g., `MATCH (n { name: 'Dvipaduttama' }) DETACH DELETE n`

Result in console:



Advance Feature of Neo4j

Cypher query language can use LOAD CSV to import data from CSV (Comma Separated Value) fileto get the data into query. The data can be loaded from standard CSV with LOAD CSV function. Firstly, the Buddhist vocabulary from the Pāli Canon was created with CSV format. From this the raw CSV data turn into a graph database which shows the nodes and the relationships between them but keeps the other details such as the meanings, PoS, and references as properties within the database. The following figure contained some words of Tipitaka and stored with CSV file format.



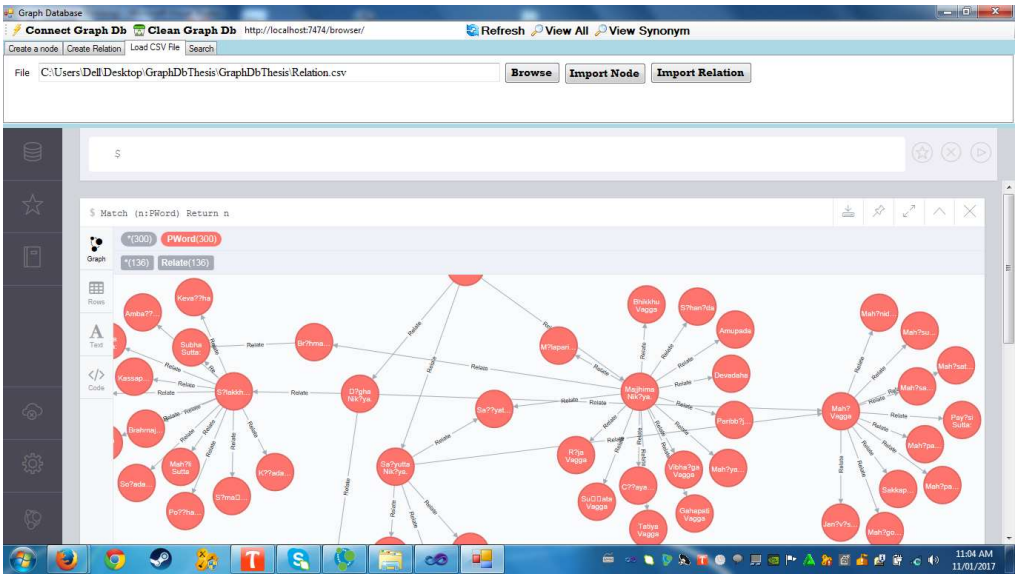
	A	B	C
2	SUB586	Uraga Sutta:	The Bhikkhu who discards all human passions (anger, hatred,craving, etc.) and is free from delusion and fear, is compared to a snake which has shed its skin.
3	SUB586	Dhaniya Sutta:	The complacent "security" of a worldling is contrasted with the genuine security of the Buddha.
4	SUB586	Khaggavisāna Sutta:	The wandering life of a Bhikkhu is praised. Family and social ties are to be avoided in view of their samsāric attachments, excepting the "good friend" (kalyāṇamitta).
5	SUB586	Kasībhāradvāja Sutta:	Socially useful or mundane labour is contrasted with the no less important efforts of the Buddha striving for Nibbāna.
6	SUB586	Cunda Sutta:	The Buddha enumerates four kinds of samanas: A Buddha, an Arahant, a conscientious Bhikkhu, a fraudulent Bhikkhu.
7	SUB586	Parābhava Sutta:	The "causes of personal downfall" in the moral and spiritual domains are enumerated.
8	SUB586	Vasala or Aggika Bhārad	In refutation of the charge "outcast," the Buddha explains that it is by actions, not lineage, that one becomes an outcast or a brahmin.
9	SUB586	Metta Sutta:	The constituents of the practice of loving-kindness towards all beings.
10	SUB586	Hemavata Sutta:	Two yakkhas have their doubts about the qualities of the Buddha resolved by him. The Buddha continues by describing the path of deliverance from death.
11	SUB586	Ālavaka Sutta:	The Buddha answers the questions of the yakkha Ālavaka concerning happiness, understanding, and the path to Nibbāna.
12	SUB586	Vijaya Sutta:	An analysis of the body into its (impure) constituent parts, and the mention of the Bhikkhu who attains Nibbāna through understanding the body's true nature.
13	SUB586	Muni Sutta:	The idealistic conception of a muni or sage who leads a solitary life freed from the passions.
14	SUB587	Ratana Sutta	A hymn to the Three Jewels: Buddha, Dhamma and Sangha
15	SUB587	Āmagandha Sutta:	Kassapa Buddha refutes the Brahmanic view of defilement through eating meat and states that this can only come about through an evil mind and corresponding actions.
16	SUB587	Hiri Sutta:	A dissertation on the nature of true friendship.
17	SUB587	Mahāmaṅgala Sutta:	Thirty-eight blessings are enumerated in leading a pure life,starting with basic ethical injunctions and culminating in the realisation of Nibbāna.
18	SUB587	Sūciloma Sutta:	In reply to the threatening attitude of the yakkha Sūciloma, the Buddha states that passion, hatred, doubt, etc., originate with the body, desire and the concept of self.
19	SUB587	Dhammacariya Sutta:	A Bhikkhu should lead a just and pure life and avoid those of a quarrelsome nature and those who are slaves of desire.
20	SUB587	Brāhmaṇadhammika Sut	The Buddha explains to some old and wealthy brahmins the high moral standards of their ancestors and how they declined, following greed for the king's wealth. As a result they ir
21	SUB587	Nava Sutta:	Taking heed of the quality of the teacher, one should go to a learned and intelligent man in order to acquire a thorough knowledge of Dhamma.
22	SUB587	Kiṃsīla Sutta:	The path of a conscientious lay disciple, Dhamma being one's first and last concern.
23	SUB587	Uṭṭhāna Sutta:	An attack on idleness and laziness. Pierced by the arrow of suffering, one should not rest until all desire is eliminated.
24	SUB587	Rāhula Sutta:	The Buddha advises his son, the novice Rāhula, to respect the wise man, associate with him, and live up to the principles of a recluse.
25	SUB587	Vaṅgīsa Sutta:	The Buddha assures Vaṅgīsa that his late teacher, Nigrodhakappa, attained Nibbāna.

Loading the data

The LOAD CSV statement can be used to load the data in from a CSV file as the following:

```
LOAD CSV WITH HEADERS FROM
C:\Users\Dell\Desktop\GraphDbThesis\GraphDbThesis\Relation.csv As
line
```

Result in console:



Classification of Pāḷi Canon

The Pāḷi Canon is the complete scripture collection of the Theravāda school. As such, it is the only set of scriptures preserved in the language of its composition. It is called the *Tipiṭaka* or "Three Baskets" because it includes the *Vinaya Piṭaka* or "Basket of Discipline," the *Sutta Piṭaka* or "Basket of Discourses," and the *Abhidhamma Piṭaka* or "Basket of Higher Teachings".

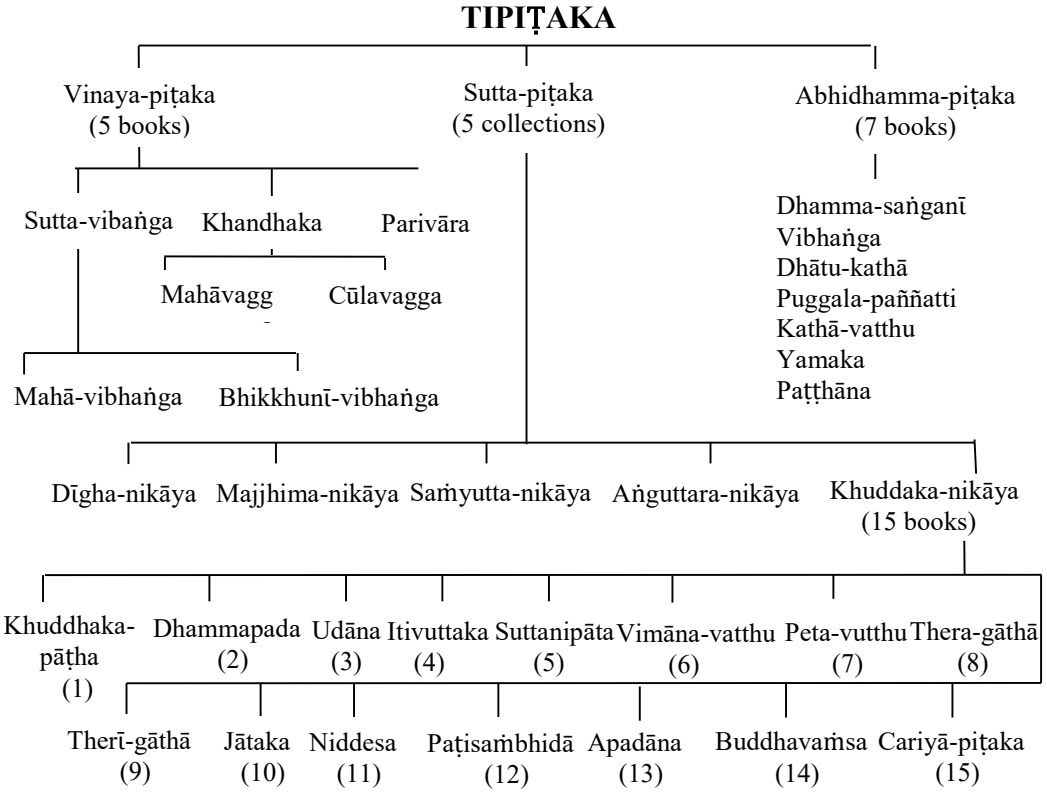


Figure 1. Classification of PāḷiCanon

Implementation for Semantic Relationship of Buddhist Vocabulary with Neo4j

Firstly, words are extracted from the Pāḷi Canon and various Buddhist literatures written by Pāḷi. The relationship implementation was focused on Pāḷi word definitions and semantic relationship in the dictionaries, where the

meaning of a word is explained by other words in its gloss.The Pāḷi words and their activities are built in the spread sheet and stored with CSV file format.The graph database extracts the words that match a user-query and sets relationships between words by using Load CSV. The user can search the desired words via graphical user interface which provides to find the words with semantic meaning. The system will display the result all of the words and its relationships with graph view. The process flow of the words and the semantic relationship of Pāḷi words implemented by Neo4j graph databasewas provided in figure 2.

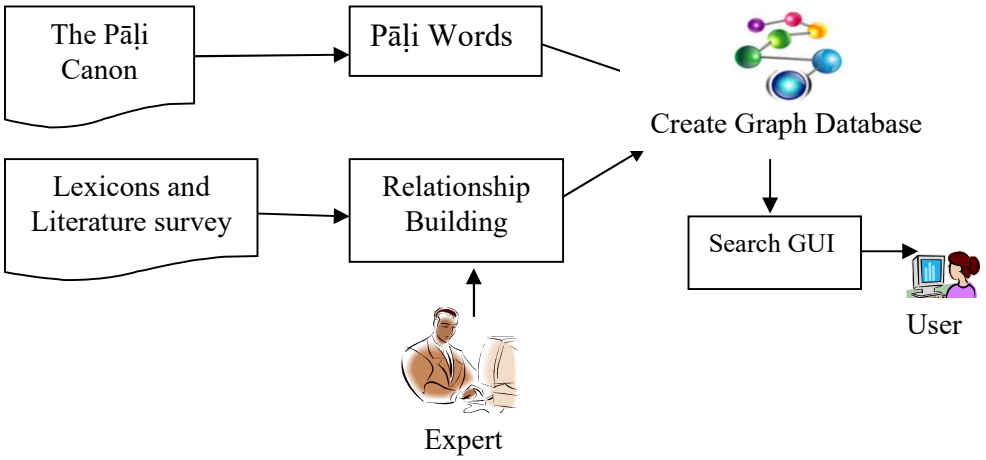


Figure 2. System architecture for Semantic Relationship between Pāḷi words

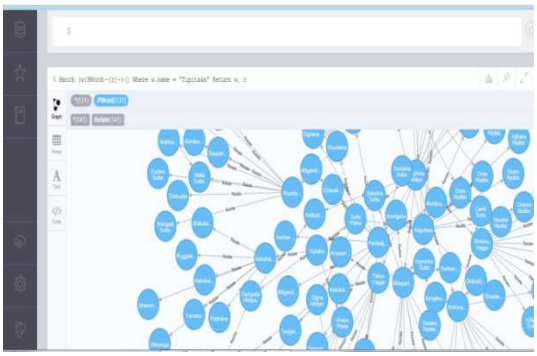


Figure (3) Relationship for Tipiṭaka

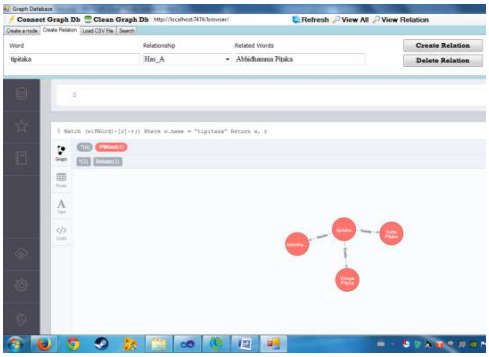


Figure (4) User Interface for Tipitaka Nodes and its Relationships

Conclusion and Future Work

In this work, a storage technique for Pāḷi Dictionary especially hyponym and meronym relationships was implemented based on graph database. Graph databases are a major pillar of the No SQL movement with lots of emerging products, such as Neo4j. Main contribute of this work is to support with Pāḷi words learner with understandable format. Yet, this is only the beginning. The automatic extraction of semantic relations of Pāḷi words form various resources will be future work. The evaluation and comparison with other graph databases and relational database were the future work. And also plan on migrating several researches done on relationship mining to work on graph database back-ends.

Acknowledgement

The author is deeply indebted to Supervisor Dr. Soe Mya Mya Aye (Professor, Head of Department, Department of Computer Studies, and University of Yangon) and Co Supervisor Dr. Khin Sandar Myint (Lecturer, Department of Computer Studies, University of Yangon) and Daw Wai Wai Myint (Lecturer and Head, Department of Computer Studies, Yadanabon University) for her encouragement. Thanks are due to Dr Thet Thet Hlaing (Lecturer, Department of Computer Studies, University of Yangon) for her helpful suggestions and valuable discussions.

References

1. Christopher S. G. Khoo and Jin-Cheon Na (2006).*Semantic Relations in Information Science.* Division of Information Studies, School of Communication &Information Nanyang Technological University.
2. http://www.neo4j.org/
3. Ian Robinson, Jim Webber, and Emil Eifrem (2013). *Graph Databases*. O'REILLY Media, Inc.,978-1-449-35626-2.
4. Justin J. Miller. Georgia Southern University, *Graph Database Applications and Concepts with Neo4j.*Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th, 2013.
5. Khin Khin Oo (2014). *The Epithets of Buddha*, Department of Oriental Studies, University of Mandalay.
6. Moggallāna, Ashin. (1982). *Abhidhānappadīpikāpaṭha*. Yangon: Department of affairs Press.
7. Murphy M. Lynne (2003). *Semantic relations and the lexicon: antonymy, synonymy, and other paradigms,* University of Sussex.
8. Renzo Angles, *A Comparison of Current Graph Database Models*, Department of Computer Science, Engineering Faculty, Universidad de Talca Camino Los Niches, Km. 1, Curic´o, Chile.
9. Rhys Davids, T. W. and Stede, W. (1979).*The Pali Text Society's Pali-English Dictionary.* London: The Pāḷi Text Society.
10. U .H. Myint, Dictionary of Pāḷi-derived Words, First Edition, Universities Press, Yangon, Myanmar, 1986.
11. Ven. Pannyavaro, *buddhist studies***:,** Buddha Dharma Education Association Inc., Buddhanet.net

COMPARATIVE ANALYSIS OF RELATIONAL AND OBJECT ORIENTED APPROACHES FOR GIS DATABASE

Yi Mon Win*

Abstract

Objects can be many things varying from an actual feature which can be extracted from a scene to more abstract entities which are associated with those features. There are a variety of database structures which can be used to store data about spatial features. These include RDBMS (Relational Database Management Systems), OODBMS (Object Oriented Database Management Systems) and ORDBMS (Object Relational Database Management Systems). All of these have retrieval systems based on SQL (Structured Query Language) and OQL (Object Query Language). The aim of this research is to compare the storage structure, retrieving data of RDBMS, ORDBMS and OODBMS storing the GIS (Geographic Information System) data of some Yangon Region's townships. This research presents a study that investigates the current scope deployment of an effective and efficient geographical information system (GIS) based approach to the representation, organization and access of these databases by Yangon Region information.

Keywords: spatial analysis, RDBMS, OODBMS, ORDBMS, SQL, GIS

Introduction

RDBMS (Relational Database Management System) and OODBMS (Object Oriented Database Management System) are both DBMSs (Database Management Systems) they differ in the model and use to represent data. OODBMSs use object-oriented model while the RDBMSs use the relational model. Both of them have their own advantages and drawbacks. OODBMS can store/ access complex data more efficiently than RDBMS. But learning OODBMS can be complex due to the object-oriented technology, compared to learning RDBMS. Therefore, choosing one over the other is dependent on the type and complexity of data that needs to be stored/ managed.

An Object-Oriented Database Management System (OODBMS), sometimes referred as Object Database Management System (ODMS) is a

*. Dr, Associate Professor and Head of Department, Department of Computer Studies, Dagon University

Database Management System (DBMS) that supports modeling and creation of data as objects. OODBMS provides support for object classes, class property and method inheritance by sub classes and their objects. A Relational Database Management System is also a DBMS but, that is based on the relational model. Most popular DBMSs currently in use are RDMSs.

Databases are traditionally used in business and administrative applications. In this research, it is discussed how these new relational databases and object oriented database can be used to solve the problems posed by spatial data management and compare database design methodologies for developing efficient schema with a spatial dimension using the GIS data as a basis. GIS considers spatial objects which can be defined in space as points, lines or areas. GIS can serve users well in its areas.

Relational Data Model for GIS data of Yangon Region

In the relational data model, information is organized in relations (two-dimensional tables). Each relation contains a set of tuples (records). Each tuple contain a number of fields. A field may contain a simple value (fixed or variable size) from some domain (e.g. integer, real, text, etc.). All of this is accomplished in the Relational DBMS through well defined terms like relation, tuple, domain, and database in Figure (1).

Township Table

Township ID	**Township Name**	**Township Map Url**
1	Ahlone	C:\Yangon Ward Area\Ahlone.shp
2	Bahan	C:\Yangon Ward Area\Bahan.shp

Township Ward Coordinate Table

TownshipDetailID	XCoordinate	Ycoordinate
1	194380.172	1857102.75
2	193364.219	1857881.75

Township Detail Table

Township DetailID	**Township**	**Population**	**Density**	**Area (Acre)**	**…**
1	Ahlone	55482	83.42	665.10	
2	Bahan	96732	53.62	1804.00	

Figure 1. Example of a Relational Data Model

ORDB (Object Relational Database) Enhanced Table Structures

An OR database consists of group of tables made up of rows. All rows in a table are structurally identical in that they all consist of a fixed number of values of specific *data types* stored in columns that are named as part of the table's definition. The most important distinction between relational tables and object-relational database tables is the way that ORDBMS columns are not limited to a standardized set of data types. Figure (2) illustrates what an object-relational table looks like.

The first thing to note about this table is the way in which its column headings consist of both a name and a data type. Second, note how several columns have internal structure. In a SQL Server DBMS, such structure would be broken up into several separate columns, and operations over a data value such as Township Name would need to list other component column in figure(3). Third, this table contains several instances of unconventional data types. X, Y Coordinate is a geographic point, which is a latitude/longitude pair that describes a position on the globe, which is a kind of Binary Large Object (BLOB) in Table (1).

Township Table

Township ID	Township Name	Township Map Url
1	Ahlone	C:\Yangon Ward Area\Ahlone.shp
2	Bahan	C:\Yangon Ward Area\Bahan.shp

Township Detail Table

Township Detail ID	Township ID	Township	Population	Density	Area (Acre)
1	1	Ahlone	55482	83.42	665.10
2	1	Bahan	96732	53.62	1804.00
. .			. .		
12	2	Kyimyindine	111514	78.38	1422.79
13	2	Mayangon	198113	31.65	6260.48

Figure 2. Inheritance in an Object-Relational Database

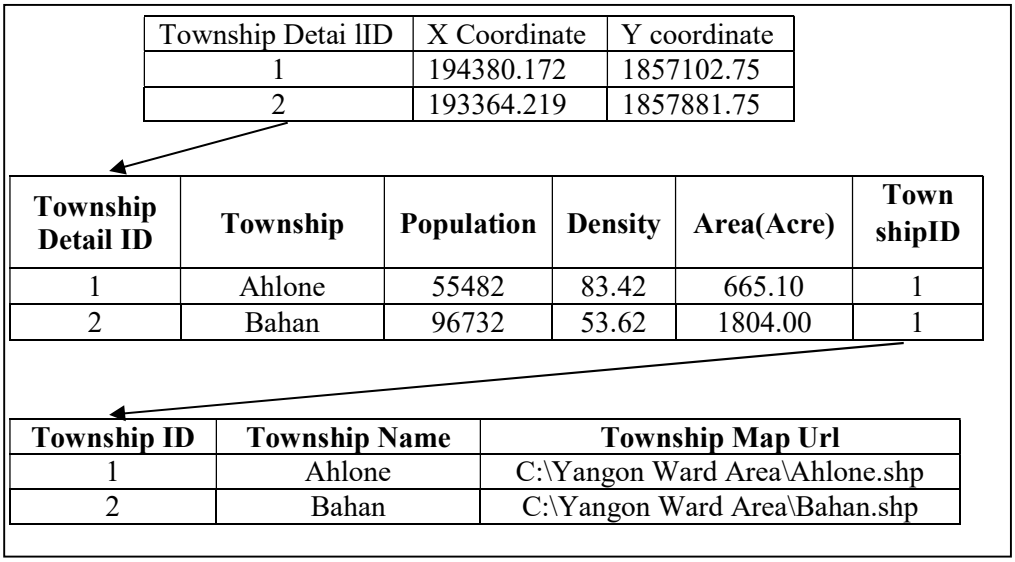


Figure 3. Object Relational Data Model

Table 1. Structure and Data for Object-Relational Table

Township ID::Township ID	**Township Name:: Ward**	**Coordinate::X,Y**
1::1	Ahlone ::Thittaw	194380.172, 1857102.75
2::2	Bahan::NgarHtatGyi(West)	193364.219, 1857881.75

Object-Oriented Data Model for GIS Data of Yangon Region

In the object-oriented data model, information is organized in graphs of objects, where each object has a number of attributes. Attributes can be simple values, complex values (part objects), references to other objects, or methods. Objects are instances of classes, and classes are (possibly) related to each by means of inheritance. The inheritance mechanism supports generalization and specialization and offers many aspects of structured reuse of models. Inheritance also offers the mechanism for qualified polymorphism, since the resulting type system can allow for objects to be recognized as belonging to several different types, namely the types of all the classes in the inheritance hierarchy which lies on the path from the instantiating class to the root of the hierarchy.

Township Table

Township ID	Township Name	Township Map Url
1	Ahlone	C:\Yangon Ward Area\Ahlone.shp
2	Bahan	C:\Yangon Ward Area\Bahan.shp

Township Detail Table

Township DetailID	Township Ward	Population (Person)	Density	Area (Acre)	Township ID
1	Ahlone	55482	83.42	665.10	1
2	Bahan	96732	53.62	1804.00	1
. .		. .			
13	Kyimyindine	111514	78.38	1422.79	2

Township Coordinate Table

Township Detail ID	X Coordinate	Y coordinate
1	194380.172	1857102.75
2	193364.219	1857881.75

Figure 4. Example of Object Oriented Table Structure for Yangon Region GIS Data

A method of an object is a specification (code) of functionality, typically manipulations of the other attributes in the same object, but may also invoke methods, associated with other objects, and thus change the state of these other objects. An important aspect of object-oriented data models is the notion of object identity: Objects has an identity (often called OID) which is totally independent of the state of the object Figure (4). That is, user can have two objects with exactly the same state (same values in all attributes), but they will still in the object system be treated as two distinct objects, with separate identities. Object modeling describes systems as built out of objects: programming abstractions that have identity, behavior, and state. Objects are an abstraction beyond abstract data types (ADTs), where data and variables are merged into a single unifying concept. As such object modeling includes many other concepts: abstraction, similarity, encapsulation, inheritance, modularity, and so on.

Experimental Results for Comparison of Databases

Experimental results are based on compare the relational, object-relational and object oriented databases by using GIS data of Yangon Region townships. Its results of comparison are comparing query processing time of these databases. Furthermore map query system compares the land use of Yangon Region. This result can apply for city planning for Yangon Region. Table (2) shows the result of query processing time and area (acre) for RDB and OODB by Yangon Region Townships.

The Comparison of Relational and Object Oriented Database by Townships

According to Table (2), the Figure (5) presents the query processing time of relational database and object oriented database on townships of Yangon Region. The

Table 2. Comparison of Query Processing Time (QPT) for RDB and OODB by Yangon Region Townships

No	Township Name	RDB (Milliseconds)	OODB (Milliseconds)	Area (Acre)	No of Records
1	Ahlone	425	214	665.60	10
2	Bahan	91	76	1804.00	22
3	Botathaung	90	64	588.80	10
4	Dagon	44	47	2880.00	4
5	Dagon Myothit (East)	60	58	6235.00	53
6	Dagon Myothit (North)	50	47	4568	27
7	Dagon Myothit Seikkan	55	45	4985.78	34
8	Dagon Myothit (South)	51	31	5096.00	36
9	Dala	57	28	2138.8	23
10	Dawbon	46	37	910.98	14
11	Hlaing Thar Yar	49	31	5699.00	28
12	Hlaing	46	29	3368	15
13	Insein	45	38	4356.23	20

No	Township Name	RDB (Milliseconds)	OODB (Milliseconds)	Area (Acre)	No of Records
14	Kamaryut	48	34	1363.2	9
15	Kyauktada	44	31	176	8
16	Kyeemyindine lower	45	40	654.23	10
17	Kyeemyindine Upper	45	40	768.56	11
18	Lanmadaw	48	32	248.9	11
19	Lathar	49	30	200.96	9
20	Mayangone	43	29	6260.48	9
21	Mingalar Taungnyut	47	34	377	19
22	Mingalardon	48	39	9875.65	33
23	North Okkalarpa	48	39	4983.35	18
24	Panbedan	44	30	187.53	12
25	Pazundaung	46	31	1056.85	9
26	Sanchaung	44	43	895.65	17
27	Seikkyi Kanaungto	46	36	1508.84	7
28	Shwepyithar	44	32	4465.32	16
29	South Okkalarpa	49	30	1900.58	13
30	Tarmwe	44	39	1184	20
31	Thakada	54	40	3215.47	18
32	Thingangyun	52	34	2841.74	38
33	Yankin	44	41	1242.89	15

system retrieves townships' attributes table from relational database and townships' map from object oriented database. The Figure (5) shows the comparison of the processing time of relational database and object oriented database. According to the result of this figure, Ahlone has more processing time than other townships because Ahlone is the first query of all townships. The Figure (6) shows the area of townships in Yangon Region. Mingalardon is the largest township and its processing time is 48 milliseconds for relational database and 39 milliseconds for object oriented database. Kyauktada Township is the smallest township in Yangon Region. Its query processing

time is 44 milliseconds for relational database and 31 milliseconds for object oriented database.

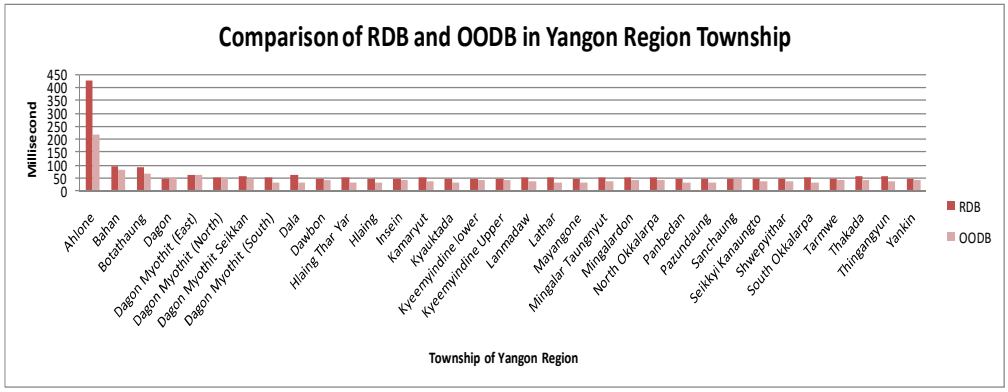


Figure 5. Comparison of Query Processing Time (QPT) for RDB and OODB by Yangon Region Townships Area

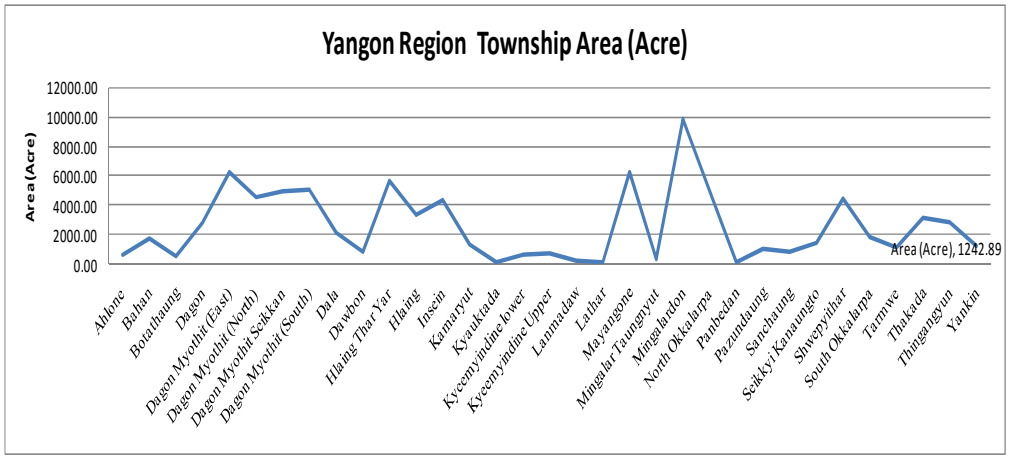


Figure 6. Comparison of Yangon Region Townships' Area

Table 3. Query Processing Time (QPT) for Comparison of ORDB and OODB by Mingalardon Township Regions

No	Region	ORDB (milliseconds)	OODB (milliseconds)	Area Meters	No of Records
1	Airport_Area	54	91.01	1032525.50	1
2	Antenna_mast_Symbol	69	92	1960.63	1
3	Benchmark_Symbol	84	153.01	2009.67	1
4	Builtup_Area	91.01	177.01	32619048.00	30
5	Bush_or_scrub Area	96.01	165.65	940709.60	6
6	Canal(Single).shp	90.01	147.01	283.87	2
7	Cemetery Area	106.01	155.01	350559.78	9
8	Cultivation Area	86	179.01	33381432.01	46
9	Dense_forest Area	87.01	159.01	2778427.75	3
10	Embankment_for_road	83	158.01	35590.91	1
11	Factory_Symbol	53	153.01	3802.18	1
12	Golf_course Area	82	176.01	2216476.75	1
13	Grass Area	61	162.01	5144866.50	13
14	Hotel_Symbol	63	162.01	12271.44	1
15	House_Building	90.01	154.01	443423.60	710
16	Intermediate_contour	73	151.01	4028148.75	63
17	Marsh_or_swamp Area	110.01	149.01	1239681.38	24
18	Monastery_Symbol	78	159.01	43115.71	26
19	Monument_Symbol	81	161.01	14076.13	7
20	Mosque_Symbol	87.01	157.01	1956.97	1
21	Open_or_barren_land_Area	89.01	153.01	17144768.00	46
22	Orchard_plantation Area	115.01	159.01	42347.00	1
23	Pagoda_or_stupa_Symbol	101.01	148.01	140256.84	38
24	Park Area	85	157.01	161752.13	2
25	Plantation Area	82	147.01	1061072.88	5
26	Police_station	89.01	153.01	352.98	2
27	Post_office	86	157.01	178.82	1
28	Public_building	110.01	167.01	1023528.94	677
29	Railway_station	101.01	151.01	15020.04	2
30	Relative_height Point	131.01	171.01	809.71	5
31	River Area	78	153.01	460.47	10
32	Scattered_trees Area	78	177.01	11575209.00	1
33	School Area	86	164.01	44105.99	6
34	Sparse_forest Area	90.01	153.01	9052389.00	22
35	Sport_field Area	86	148.01	62010.20	59
36	Supplementary_contour	113.01	172.01	15449770.11	5
38	Triangulationstat_Symbol	85	166.01	1352580.88	7
39	Vegitation_boundary	91.01	159.01	1352580.88	7
40	Cemetery Area	90.01	151.01	8423.03	1
41	Lake_or_pond	75	156.01	3535622.50	206

Comparison of Object Relational and Object Oriented Database by Townships

According to Table (3), the Figure (7) shows the query processing time of the maps of Mingalardon township's regions from both databases. Mingalardon is the largest township and has multiple regions in Yangon Region among 33 townships. The Figure (8) shows the regions of Mingalardon Township. Cultivation region is the largest region and its area is 33381432.01 square meters. The Figure (7) shows query processing time where object relation database is 86 milliseconds and object oriented database is 179 milliseconds. The smallest region is the post office region (178.82 square meters) and its query processing time of object relational database is 81 milliseconds and object oriented database is 157.01milliseconds.

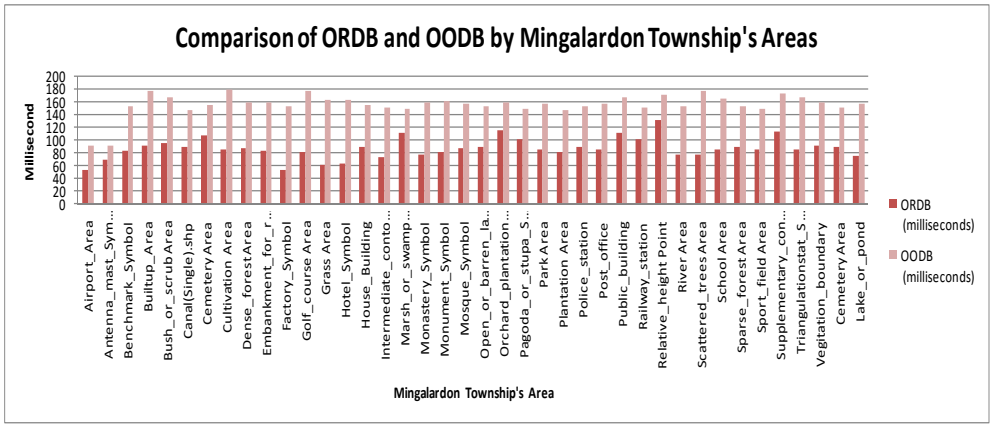


Figure 7. Comparison of QPT for ORDB and OODB by Mingalardon Township's Regions

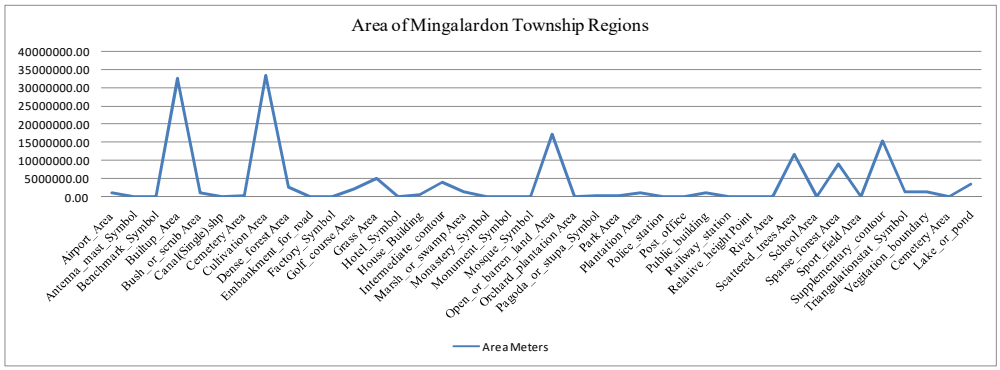


Figure 8. Area of Mingalardon Township's Regions

Conclusion

This research is comparing relational database, object relational database and object oriented database for GIS data of 33 townships in Yangon Region by using map query system. This map query system has been implemented by using Microsoft C#.Net 2010, MapWin GIS, MS SQL server 2010 for object relational database, db4o database for object oriented database on Intel® Dual Core CPU P6100 @ 2.00 GB main memory and Microsoft Window7 Ultimate. The experimental results are taken out from these computer specifications. It can vary depending on the enhancement of computer specifications. According to the result and discussion, it is generally concluded that the effective use of structured query language (SQL) on sql server 2010 for ORDB and query by example method on db4o database for OODB. It analyses the performance of the different query languages and same sizes of different databases.

This system is compared to the query processing time performance of RDB, ORDB and OODB. It is noticed that the experimental results in the figure and tables are counted from the outcome of the first time running on the query processing. All results are taken from the results of the first time query processing not from the result of the next times because same query processing are faster than the first time. This is because the database optimizer optimizes execution by using least recently use (LRU) algorithm for frequents the same query.

As a result of comparison of relational and object oriented database, the system retrieves the township attribute data tables from relational database that is more processing time than OODB and retrieves GIS township map from object oriented database that is less processing time than RDB because the traditional RDBMSs are not suitable for applications with complex data structures or new data types for large, unstructured objects, such as CAD/ CAM, Geographic information systems, multimedia databases, imaging and graphics.

According to comparison of object relation and object oriented database by townships, the Mingalardon township and Kyauktada township are compared by object relational and object oriented database. In the result of this comparion, Mingalardon is the biggest township and Kyauktada township

is the smallest township of Yangon Region. Mingalardon township owns 41 regions and Kyauktada owns 13 regions. Query processing time of ORDB is faster than OODB because of ORDB employs object-oriented concepts and capabilities on top of a conventional relational database management system (RDBMS). ORDBMSs are extensions of RDBMSs. The ORDBMS standard SQL: 1999 is a superset of the purely relational SQL-92 standard. Hence, all relational features are still available in ORDBMSs.

According to comparison of object relational and object oriented database by each region of townships, query processing time of object oriented database has more processing time than object relational database. First compared region is built up region; every townships of Yangon Region has built up region. Mingalardon has the largest built up area and Lathar owns smallest built up area. Second largest built up area is Shwe Pyi Thar and the second smallest is Panbedan township.

This research work performs well on the comparing relational database, object relational database and object oriented database for GIS data of 33 townships in Yangon Region. It also supports well for understanding how to build the databases and to retrieve from these database by using query languages. This research analyses on various types of databases and their query languages about structured query language in SQL server 2010 and object query language in db4o. In this research, Object Relational database is the more effective than other databases. Therefore one of the future works is to extend this research will build the data center for three dimensional urban planning of Yangon Region GIS data, and will retrieve these data by using oracle spatial query language. So, the query performance will view different urban land use pattern with three dimensions. Furthermore, the system can apply city planning of Yangon Region land use.

Acknowledgement

I wish to thank Dr Win Naing, Rector of Dagon University and Dr Aye Aye Tun and Dr Nu Nu Yee, Pro Rector of Dagon University for permitting the opportunity to present this paper. The author wishes to express her gratitude to professor (Retired) Dr Nwe Nwe Win, Head of Department of Computer Studies for her encouragement to carry out this presentation.

References

1. Agathoniki, T., Semantic optimization of OQL queries, University of Cambridge. 2002
2. Bergholt, L DTI et al, Database Management Systems: Relational, Object-Relational, and Object-Oriented Data Models, COT/4-02-V1.1.C. Feb 25,1998
3. Date, C.J, Foundation for Future Database Systems: The Third Manifesto,

   2nd edition, Addison-Wesley Professional, 2000
4. Department of Population, Ministry of Immigration and Population, May-2015, The 2014 Myanmar Population and Housing Census, The Union Report
5. Kiran Kumar, B.V.S.S.D et al., DO-GIS - a distributed and object oriented GIS, 2012
6. Worboys & Hearnshaw etal, Object-oriented Data Modelling for Spatial Databases, International Journal of Geographical Information Systems, Volume 4, Number 4. 1990

MASSIVELY PARALLEL POPULATION-BASED MONTE CARLO METHODS WITH MANY-CORE PROCESSORS

Wint Pa Pa Kyaw*

Abstract

This research presents the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. Graphics cards, containing multiple Graphics Processing Units (GPUs), are self-contained parallel computational devices that can be housed in conventional desktop and laptop computers and can be thought of as prototypes of the next generation of many-core processors. For certain classes of population-based Monte Carlo (MC) algorithms they offer massively parallel simulation, with the added advantage over conventional distributed multi-core processors that they are cheap, easily accessible, easy to maintain, easy to code, dedicated local devices with low power consumption. On a canonical set of stochastic simulation examples including population-based Markov chain Monte Carlo (MCMC) methods and Sequential Monte Carlo (SMC) methods, speedups are found from 35 to 500 fold over conventional single-threaded computer code. These findings suggest that GPUs have the potential to facilitate the growth of statistical modelling into complex data rich domains through the availability of cheap and accessible many-core computation.

Keywords: Sequential Monte Carlo, Population-Based Markov Chain Monte Carlo, General Purpose Computation on Graphics Processing Units, Many-Core Architecture, Stochastic Simulation, Parallel Processing

Introduction

This research describes the utility of graphics cards involving Graphics Processing Units (GPUs) to perform local, dedicated, massively parallel stochastic simulation. GPUs were originally developed as dedicated devices to aid in real-time graphics rendering. However recently there has been an emerging literature on their use for scientific computing as they house multicore processors. Many advanced population-based Monte Carlo (MC) algorithms are ideally suited to GPU simulation and offer significant speed up over single CPU implementation. The focus is on the parallelization of general

*. Dr., Associate Professor, Department of Computer Studies, University of Yangon

sampling methods. Moreover, this research shows how the choice of population-based MC algorithm for a particular problem can depend on whether one is running the algorithm on a GPU or a CPU.

To gain an understanding of the potential benefits to statisticians this research has investigated speedups on a canonical set of examples taken from the population-based MC literature. These include Bayesian inference for a Gaussian mixture model computed using a population-based Markov Chain Monte Carlo (MCMC) method. The idea of splitting the computational effort of parallelizable algorithms amongst processors is certainly not new to statisticians. In fact, distributed systems and clusters of computers have been around for decades. Many-core processor communication has very low latency and very high bandwidth due to high-speed memory that is shared amongst the cores. Low latency here means the time for a unit of data to be accessed or written to memory by a processor is low while high bandwidth means that the amount of data that can be sent in a unit of time is high. For many algorithms, this makes parallelization viable where it previously was not. In addition, the energy efficiency of a many-core computation compared to a single-core or distributed computation can be improved. This is because the computation can both take less time and require less overhead. Finally, these features enable the use of parallel computing for researchers outside traditional high-cost centers housing high-performance computing clusters.

The speedup is chosen to investigate for the simulation of random variates from complex distributions, a common computational task when performing inference using MC. In particular, population-based MCMC methods and SMC methods are focused on for producing random variates as these are not algorithms that typically see significant speedup on clusters due to the need for frequent, high-volume communication between computing nodes. This work focuses on the suitability of many-core computation for MC algorithms whose structure is parallel, since this is of broad theoretical interest, as opposed to a focusing on parallel computation of application-specific likelihoods.

The algorithms are implemented for the Compute Unified Device Architecture (CUDA) and make use of GPUs which support this architecture. CUDA offers a fairly mature development environment via an extension to the C programming language. For applications CUDA version 5.5 with an

NVIDIA GT 750M are used. The GT 750M has 384 multiprocessors. For all current NVIDIA cards, a multiprocessor comprises 8 arithmetic logic units (ALUs), 2 special units for transcendental functions, a multithreaded instruction unit and on-chip shared memory. For example, for single-precision floating point computation, one can think of the GT 750 as having 3072 (384 × 8) single processors. The current generation of GPUs is 4-8 times faster at single precision arithmetic than double precision. Single precision seems perfectly sufficient for the applications in this research since the variance of the Monte Carlo estimates exceeds the perturbations due to finite machine precision.

Graphics Processing Unit for Parallel Processing

GPUs have evolved into many-core processing units, currently with up to 30 multiprocessors per card, in response to commercial demand for real-time graphics rendering, independently of demand for many-core processors in the scientific computing community. As such, the architecture of GPUs is very different to that of conventional CPUs. An important difference is that GPUs devote proportionally more transistors to ALUs and less to caches and flow control in comparison to CPUs. This makes them less general purpose but highly effective for data-parallel computation with high arithmetic intensity, i.e. computations where the same instructions are executed on different data elements and where the ratio of arithmetic operations to memory operations is high. This Single Instruction Multiple Data (SIMD) architecture puts a heavy restriction on the types of computation that optimally utilize the GPU but in cases where the architecture is suitable it reduces overhead.

Figure 1 gives a visualization of the link between a host machine and the graphics card, emphasizing the data bandwidth characteristics of the links and the number of processing cores. A program utilizing a GPU is hosted on a CPU with both the CPU and the GPU having their own memory. Data is passed between the host and the device via a standard memory bus, similar to how data is passed between main memory and the CPU. The memory bus between GPU memory and the GPU cores is both wider and has a higher clock rate than a standard bus, enabling much more data to be sent to the cores than the equivalent link on the host allows. This type of architecture is ideally

suited to data-parallel computation since large quantities of data can be loaded into registers for the cores to process in parallel. In contrast, typical computer architectures use a cache to speed up memory accesses using locality principles that are generally good but do not fully apply to data-parallel computations, with the absence of temporal locality most notable.

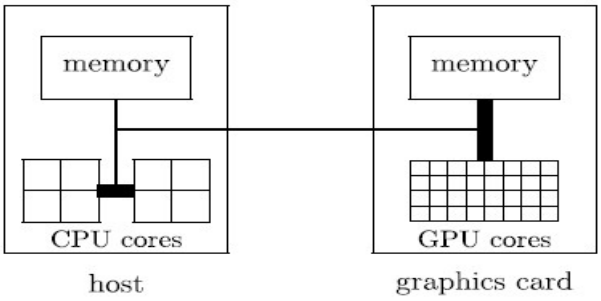


Figure 1: Link between host and graphics card. The thicker lines represent higher data bandwidth while the squares represent processor cores.

Graphics Processing Units Parallelizable Algorithms

In general, if a computing task is well-suited to SIMD parallelization then it will be well-suited to computation on a GPU. In particular, data-parallel computations with high arithmetic intensity (computations where the ratio of arithmetic operations to memory operations is high) are able to attain maximum performance from a GPU. This is because the volume of very fast arithmetic instructions can hide the relatively slow memory accesses. It is crucial to determine whether a particular computation is data-parallel on the instruction level when determining suitability. From a statistical simulation perspective, integration via classical Monte Carlo or importance sampling is ideal computational tasks in a SIMD framework. This is because each computing node can produce and weight a sample in parallel, assuming that the sampling procedure and the weighting procedure have no conditional branches. If these methods do branch, speedup can be compromised by many computing nodes running idle while others finish their tasks. This can occur, for example, if the sampling procedure uses rejection sampling.

In contrast, if a computing task is not well-suited to SIMD parallelization then it will not be well-suited to computation on a GPU. In

particular, task-parallel computations where one executes different instructions on the same or different data cannot utilize the shared flow control hardware on a GPU and often end up running sequentially. Even when a computation is data-parallel, it might not give large performance improvements on a GPU due to memory constraints. This can be due to the number of registers required by each thread or due to the size and structure of the data necessary for the computation requiring large amounts of memory to be transferred between the host and the graphics card.

Many statistical algorithms involve large data sets, and the extent to which many-core architectures can provide speedup depends largely on the types of operations that need to be performed on the data. For example, many matrix operations derive little speedup from parallelization except in special cases, e.g. when the matrices involved are sparse. It is difficult to classify concisely the types of computations amenable to parallelization beyond the need for data-parallel operations with high arithmetic intensity. However, experience with parallel computing should allow such classifications to be made prior to implementation in most cases.

Parallelizable Sampling Methods

A number of sampling methods for parallel implementations can be produced without significant modification. There is an abundance of statistical problems that are essentially computational in nature, especially in Bayesian inference. In many such cases, the problem can be distilled into one of sampling from a probability distribution whose density π, pointwise and up to a normalizing constant can be computed, that is, $\pi^*(\cdot)$ where $\pi(x) = \pi^*(x)/Z$ can be computed. A common motivation for wanting samples from π is so expectations of certain functions can be computed. If such a function is denoted by ϕ, the expectation of interest is

$$I \triangleq \int_{x \in \mathcal{X}} \phi(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}$$

The Monte Carlo estimate of this quantity is given by

$$\hat{I}_{MC} \triangleq \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}^{(i)})$$

where $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ are samples from π.

Samples from π in order to compute this estimate are needed. In practice, one often cannot sample from π directly. There are two general classes of methods for dealing with this. The first are importance sampling methods, where the weighted samples are generated from π by generating N samples according to some importance density γ proportional to γ^* and then estimating I via

$$\hat{I}_{IS} \triangleq \sum_{i=1}^{N} W^{(i)}\phi(\mathbf{x}^{(i)})$$

where $W^{(i)}$ are normalized importance weights

$$W^{(i)} = \frac{w(\mathbf{x}^{(i)})}{\sum_{j=1}^{N} w(\mathbf{x}^{(j)})} \text{ and } w(\mathbf{x}^{(i)}) = \frac{\pi^*(\mathbf{x}^{(i)})}{\gamma^*(\mathbf{x}^{(i)})}$$

The asymptotic variance of this estimate is given by $C(\phi, \pi, \gamma)/N$, that is, a constant over N. For many problems, it is difficult to come up with an importance density γ such that $C(\phi, \pi, \gamma)$ is small enough to attain reasonable variance with practical values of N.

The second general class of methods are MCMC methods, in which an ergodic π-stationary Markov chain is sequentially constructed. Once the chain has converged, all the dependent samples can be used to estimate I. The major issue with MCMC methods is that their convergence rate can be prohibitively slow in some applications.

For example, naive importance sampling, like classical Monte Carlo, is intrinsically parallel. Therefore, in applications where one have access to a good importance density γ, linear speedup can be got with the number of processors available. Similarly, in cases where MCMC converges rapidly, the estimation of I can be parallelized by running separate chains on each processor. While these situations are hoped for, they are not particularly interesting from a parallel architecture standpoint because they can run

equally well in a distributed system. Finally, this research is not concerned with problems for which the computation of individual MCMC moves or importance weights are very expensive but themselves parallelizable. While the increased availability of parallel architectures will almost certainly be of help in such cases, the focus here is on potential speedups by parallelizing general sampling methods. Example of recent work in this area can be found in this research, in which speedup is obtained by parallelizing evaluation of individual likelihoods.

Population-Based Markov chain Monte Carlo

A common technique in facilitating sampling from a complex distribution π with support in X is to introduce an auxiliary variable $a \in A$ and sample from a higher-dimensional distribution $\bar{\pi}$ with support in the joint space $A \times X$, such that $\bar{\pi}$ admits π as a marginal distribution. With such samples, one can discard the auxiliary variables and be left with samples from π. A kernel will generally refer to a Markov chain transition kernel as opposed to a CUDA kernel.

This idea is utilized in population-based MCMC, which attempts to speed up convergence of an MCMC chain for π by instead constructing a Markov chain on a joint space X^M using $M - 1$ auxiliary variables each in X. In general, one have M parallel 'subchains' each with stationary distribution $\pi_i, i \in \mathcal{M} \triangleq \{1, \ldots, M\}$ and $\pi_M = \pi$. Associated with each subchain i is an MCMC kernel L_i that leaves π_i invariant, and which one run at every time step. Of course, without any further moves, the stationary distribution of the joint chain is

$$\bar{\pi}(\mathbf{x}_{1:M}) \triangleq \prod_{i=1}^{M} \pi_i(\mathbf{x}_i)$$

and so if $\mathbf{x}_{1:M} \sim \bar{\pi}$, then $\mathbf{x}_M \sim \pi$. This scheme does not affect the convergence rate of the independent chain M. However, since mixtures of $\bar{\pi}$-stationary MCMC kernels can be cycled without affecting the stationary distribution of the joint chain, certain types of interaction between the subchains can be allowed which can speed up convergence. In general, a series of MCMC kernels that act on subsets of the variables is applied. The number of second-stage MCMC kernels are denoted by R and the MCMC kernels themselves as $K_1, \ldots, K_R$, where kernel K_j operates on variables with indices in $I_j \subset M$. The

idea is that the R kernels are executed sequentially and it is required that each K_j leave $\Pi_{I \in Ij} \pi_i$ invariant.

Given π, there are a wide variety of possible choices for M, $\pi_{1:M-1}$, $L_{1:M}$, R, $I_{1:R}$ and $K_{1:R}$ which will affect the convergence rate of the joint chain. The first stage of moves involving $L_{1:M}$ is trivially parallelizable. However, the second stage is sequential in nature. For a parallel implementation, it is beneficial for the I_j's to be disjoint as this allows the sequence of exchange kernels to be run in parallel. Of course, this implies that $I_{1:R}$ should vary with time since otherwise there will be no interaction between the disjoint subsets of chains. Furthermore, if the parallel architecture used is SIMD (Single Instruction Multiple Data) in nature, it is desirable to have the K_j's be nearly identical algorithmically. The last consideration for parallelization is that while speedup is generally larger when more computational threads can be run in parallel, it is not always helpful to increase M arbitrarily as this can affect the convergence rate of the chain. However, in situations where a suitable choice of M is dwarfed by the number of computational threads available, one can always increase the number of chains with target π to produce more samples.

Population-Based MCMC Algorithm

There are two types of moves:

1. In parallel, each chain i performs an MCMC move targetting π_i.
2. In parallel, adjacent chains i and $i + 1$ perform an MCMC 'exchange' move targeting $\pi_i \pi_{i+1}$

A simple exchange move at time n proposes to swap the values of the two chains and has acceptance probability

$$\min\{1, \frac{\pi_i(x_{i+1}^{(n)})\pi_{i+1}(x_i^{(n)})}{\pi_i(x_i^{(n)})\pi_{i+1}(x_{i+1}^{(n)})}\}.$$

In order to ensure (indirect) communication between all the chains, the exchange partners are picked at each time with equal probability from

$$\{\{1,2\},\ldots,\{M-1,M\}\} \text{ and } \{\{2,3\},\ldots,\{M-2,M-1\}\}.$$

Sequential Monte Carlo Samplers

SMC samplers are a more general class of methods that utilize a sequence of auxiliary distributions $\pi_0, \ldots, \pi_T$, much like population-based MCMC. However, in contrast to population-based MCMC, SMC samplers start from an auxiliary distribution π_0 and recursively approximate each intermediate distribution in turn until finally $\pi_T = \pi$ is approximated. The algorithm has the same general structure as classical SMC, with differences only in the types of proposal distributions, target distributions and weighting functions used in the algorithm.

The difference between population-based MCMC and SMC samplers is subtle but practically important. Both can be viewed as population-based methods on a similarly defined joint space since many samples are generated at each time step in parallel. However, in population-based MCMC the samples generated at each time each have different stationary distributions and the samples from a particular chain over time provide an empirical approximation of that chain's target distribution. In SMC samplers, the weighted samples generated at each time approximate one auxiliary target distribution and the true target distribution is approximated at the last time step.

Algorithmic Details

1. At time t = 0:

   For i = 1,…,N, sample $\mathbf{x}_0^{(i)} \sim \eta(\mathbf{x}_0)$

   For i = 1,…,N, evaluate the importance weights:

$$w_0(\mathbf{x}_0^{(i)}) \propto \frac{\pi_0(\mathbf{x}_0^{(i)})}{\eta(\mathbf{x}_0^{(i)})}$$

2. For times t = 1,…, T:

   For i = 1,…,N, sample

$$\mathbf{x}_t^{(i)} \sim K_t(\mathbf{x}_{t-1}^{(i)}, \cdot).$$

For i = 1,…,N, evaluate the importance weights:

$$w_t(\mathbf{x}_t^{(i)}) \propto w_{t-1}(\mathbf{x}_{t-1}^{(i)}) \frac{\pi_t(\mathbf{x}_t^{(i)}) L_{t-1}(\mathbf{x}_t^{(i)}, \mathbf{x}_{t-1}^{(i)})}{\pi_{t-1}(\mathbf{x}_{t-1}^{(i)}) K_t(\mathbf{x}_{t-1}^{(i)}, \mathbf{x}_t^{(i)})}.$$

Normalize the importance weights. Depending on some criteria, resample the particles. Set

$$w_t^{(i)} = \frac{1}{N} \text{ for } i = 1, \ldots, N.$$

For the special case where L_{t-1} is the associated backwards kernel for K_t , ie.

$$\pi_t(x_t) L_{t-1}(x_t, x_{t-1}) = \pi_t(x_{t-1}) K_t(x_{t-1}, x_t).$$

the incremental importance weights simplify to

$$w_t(\mathbf{x}_t^{(i)}) \propto w_{t-1}(\mathbf{x}_{t-1}^{(i)}) \frac{\pi_t(\mathbf{x}_{t-1}^{(i)})}{\pi_{t-1}(\mathbf{x}_{t-1}^{(i)})}.$$

The normalization step is a reduction operation and a divide operation. The resampling step involves a parallel scan.

Implementation of Canonical Examples

To demonstrate the types of speed increase one can attain by utilizing GPUs, each method to a representative statistical problem is applied. Bayesian inference for a Gaussian mixture model is used as an application of the population-based MCMC and SMC samplers.

The applications are representative of the types of problems that these methods are commonly used to solve. In particular, while the distribution of mixture means given observations is only one example of a multimodal distribution, it can be thought of as a canonical distribution with multiple well-separated modes. Therefore, the ability to sample points from this distribution is indicative of the ability to sample points from a wide range of multimodal distributions.

Mixture Modeling

Finite mixture models are a very popular class of statistical models as they provide a flexible way to model heterogeneous data. Let $\mathbf{y} = y_{1:m}$ denote identically independent distribution (i.i.d) observations where $y_j \in R$ for $j \in \{1, \ldots, m\}$. A univariate Gaussian mixture model with k components states that each observation is distributed according to the mixture density

$$p(y_j|\mu_{1:k}, \sigma_{1:k}, w_{1:k-1}) = \sum_{i=1}^{k} w_i f(y_j|\mu_i, \sigma_i),$$

where f denotes the density of the univariate normal distribution. The density of $\boldsymbol{y}$ is then equal to

$$\prod_{j=1}^{m} p(y_j|\mu_{1:k}, \sigma_{1:k}, w_{1:k-1}).$$

For simplicity, assume that k, $w_{1:k-1}$ and $\sigma_{1:k}$ are known and that the prior distribution on μ is uniform on the k-dimensional hypercube $[-10, 10]^k$. $k = 4$, $\sigma_i = \sigma = 0.55$, $w_i = w = 1/k$ for $i \in \{1, \ldots, k\}$ are set. $m = 100$ observations are simulated for $\mu = \mu_{1:4} = (-3, 0, 3, 6)$. The resulting posterior distribution for μ is given by

$$p(\boldsymbol{\mu}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\mu})\mathbb{I}(\boldsymbol{\mu} \in [-10, 10]^4).$$

The main computational challenge associated with Bayesian inference in finite mixture models is the nonidentifiability of the components. As exchangeable priors have been used for the parameters $\mu_{1:4}$, the posterior distribution $p(\mu|y)$ is invariant to permutations in the labeling of the parameters. Hence this posterior admits $k! = 24$ symmetric modes, which basic random-walk MCMC and importance sampling methods typically fail to characterize using practical amounts of computation. Generating samples from this type of posterior is a popular method for determining the ability of samplers to explore a high-dimensional space with multiple well-separated modes.

Population-Based Markov chain Monte Carlo

The auxiliary distributions $\pi_{1:M-1}$ following the parallel tempering methodology are selected, that is, $\pi_i(\mathbf{x}) \propto \pi(\mathbf{x})^{\beta_i}$ with $0 < \beta_1 < \ldots < \beta_M = 1$ and use $M = 200$. This class of auxiliary distributions is motivated by the fact that

MCMC converges more rapidly when the target distribution is flatter. For this problem, the cooling schedule $\beta_i = (i/M)^2$ and a standard N(0,Ik) random walk Metropolis-Hastings kernel are used for the first stage moves.

For the second stage moves, the basic exchange move are used, chains i and j swap their values with probability min{1, α_{ij}} where

$$\alpha_{ij} = \frac{\pi_i(\mathbf{x}_j)\pi_j(\mathbf{x}_i)}{\pi_i(\mathbf{x}_i)\pi_j(\mathbf{x}_j)}.$$

Further, the exchanges to take place only between adjacent chains are allowed so that all moves can be done in parallel. $R = M/2$ and I1:R is either {{1, 2}, {3, 4}, … , {M − 1, M}} or {{2, 3}, {4, 5}, … , {M – 2, M − 1}, {M, 1}}, each with probability half are used. Emphasize that all first stage MCMC moves are executed in parallel on the GPU, followed by all the exchange moves being executed in parallel. The following code segments are to get compute value function properties for MCMC.

```
void mcmc(int M, int nb, int nt)
{
        generate_mix_data(k, sigma, mus, data_array, L);
        compute_ci1_ci2(sigma, 1.0f / k, c1, c2);
        populate_rand_d(d_array_init, numChains * k);
        multiply(numChains * k, d_array_init, d_array_init, 20, nb, nt);
        add(numChains * k, d_array_init, d_array_init, -10, nb, nt);
}
```

To test the computational time required by the algorithms the number of chains are allowed to vary but fix the number of points which wishing to sample from the marginal density $\pi_M = \pi$ at 8192. As such, an increase in the number of chains leads to a proportional increase in the total number of points sampled.

Sequential Monte Carlo Sampler

As with population-based MCMC, a tempering approach and the same cooling schedule are used, this is, $\pi_t(\mathrm{x}) \propto \pi(\mathrm{x})^{\beta_t}$ with $\beta_t = (t/M)^2$ and $M = 200$. The uniform prior on the hypercube are used to generate the samples $\{\mathrm{x}_0^{(1:N)}\}$ and perform 10 MCMC steps with the standard $N(0,I_k)$ random walk Metropolis-Hastings kernel at every time step. The generic backward kernel is

used for the case where each kernel is π_t -stationary so that the unnormalized incremental importance weights are of the form $\pi_t(\mathbf{x}_{t-1})/\pi_{t-1}(\mathbf{x}_{t-1})$. The following code segments are to compute value function properties for SMCS.

```
void testMG(int N, int nb, int nt)
 {
      generate_mix_data(k, sigma, mus, data_array, L);
      compute_ci1_ci2(sigma, 1.0f / k, c1, c2);
      populate_rand_d(d_array_init, N * k);
      multiply(N * k, d_array_init, d_array_init, 20, nb, nt);
      add(N * k, d_array_init, d_array_init, -10, nb, nt);
      testMG(N, k, T, numSteps, d_array_init, temps, h_args_t1, h_args_t2, nb, nt);
      testMG_host(N, k, T, numSteps, array_init, temps, h_args_t1, h_args_t2);
 }
```

Results and Discussion

The parallel code is run on a computer equipped with an NVIDIA GT 750M GPU, and the reference single-threaded code is run on a Intel (R)core(TM)i7 4500U CPU 1.80GHz processor. The resulting processing times and speedups are given in Tables 1–2.

Population-Based Markov chain Monte Carlo Results

Table 1: Running times for the Population-Based MCMC Sampler for various numbers of chains M.

N = 8192 points are sampled from chain M.

M	CPU(secs)	GT 750 M(secs)	Speedup
(1) 8	1.33	0.93	1
(2) 32	5.32	1.03	5
(3) 128	20.00	1.89	11
(4) 512	62.40	1.24	50
(5)2048	249.64	1.43	175
(6)8192	998.42	2.32	430
(7)32768	4002.00	7.73	518
(8)131072	16218.00	28.35	572

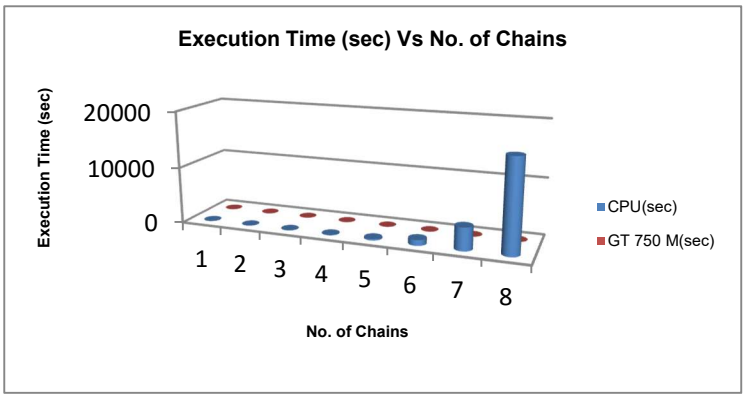


Figure 2: The relation of execution time and number of chains

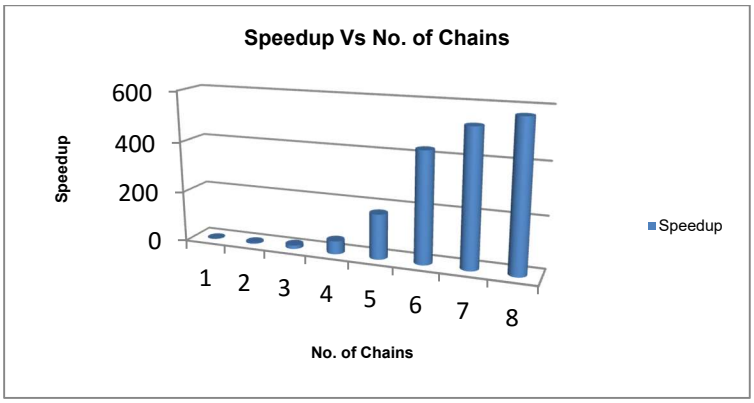


Figure 3: The relation of speedup and number of chains

Processing times for MCMC code are given in Table 1, in which one can see that using 131072 chains is impractical on the CPU but entirely reasonable using the GPU. Figure 2 shows that GPU time is faster than CPU time. Figure 3 shows that speedup goes faster with increasing the number of chains. So it can be observed that parallel computing is more suitable for enormous data.

Sequential Monte Carlo Sampler Results

Table 2: Running times for the Sequential Monte Carlo Sampler for various values of N.

N	CPU(secs)	GT 750 M (secs)	Speedup
(1)8192	266.40	0.60	444
(2)16384	529.20	1.11	477
(3)32768	1062.00	2.19	485
(4)65536	2118.00	4.50	471
(5)131072	4236.00	8.08	524
(6)262144	8460.00	16.22	522

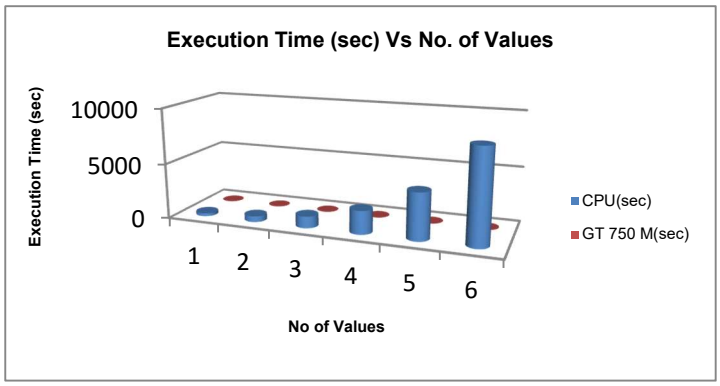


Figure 4: The relation of execution time and number of values

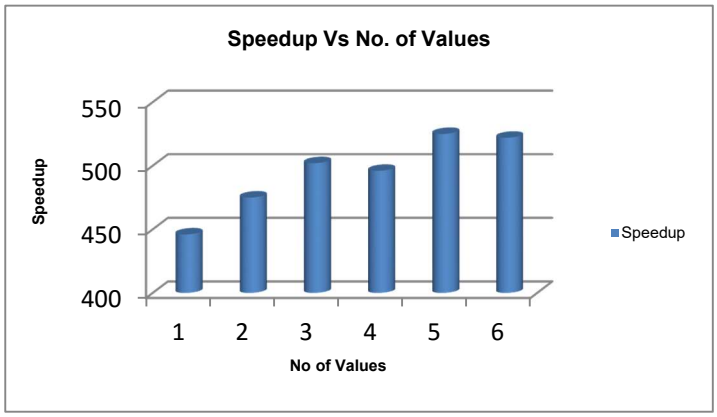


Figure 5: The relation of speedup and number of values

Processing times for SMCS code are given in Table 2. GPU execution time is faster than CPU execution time in SMC sampler that shown in Figure 4. Figure 5 shows that speedup goes faster with increasing the number of values.

Discussion

The speedup for the population-based MCMC algorithm and the SMC sampler is tremendous. In particular, the evaluation of $p(y|\mu)$ for the mixture-modelling application has high arithmetic intensity since it consists of a product-sum operation with 400 Gaussian log-likelihood evaluations involving only 104 values. In fact, because of the low register and memory requirements, so many threads can be run concurrently that SIMD calculation of this likelihood can be sped up by 500 times on the GT 750M. Estimation of static parameters in continuous state-space models or the use of SMC proposals within MCMC can require thousands of runs, so a speedup of this scale can substantially reduce the computation time of such approaches. Speedups can be expected in the vicinity of 500 with SMC if few resampling steps are required and each weighting step has small space complexity and moderate time complexity.

While CUDA have been used to implement the parallel components of algorithms, the results are not necessarily specific to this framework or to GPUs. It is expected that the many-core processor market will grow and there will be a variety of different devices and architectures to take advantage of. However, the SIMD architecture and the sacrifice of caching and flow control for arithmetic processing is likely to remain since when it is well-suited to a problem it will nearly always deliver considerable speedup. For users who would like to see moderate speedup with very little effort, there is work being done to develop libraries that will take existing code and automatically generate code that will run on a GPU.

The speedups attainable with many-core architectures have broad implications in the design, analysis, and application of SMC and population-based MCMC methods. In application, this does not occur until one have around 4096 auxiliary distributions. One might notice that this number is far larger than the number of processors on the GPU. This is due to the fact that

even with many processors, significant speedup can be attained by having a full pipeline of instructions on each processor to hide the relatively slow memory reads and writes. In both SMC and MCMC, it is also clear from this case study that it is beneficial for each thread to use as few registers as possible since this determines the number of threads that can be run simultaneously. This may be of interest to the methodology community since it creates a space-time trade-off that might be exploited in some applications.

A consequence of the space-time trade-off mentioned above is that methods which require large numbers of registers per thread are not necessarily suitable for parallelization using GPUs. For example, operations on large, dense matrices that are unique to each thread can restrict the number of threads that can run in parallel and hence dramatically affect potential speedup. In cases where data are shared across threads, however, this is not an issue. In principle, it is not the size of the data that matters but the space complexity of the algorithm in each thread that dictates how scalable the parallelization is.

Conclusion

The potential of parallel processing to aid in statistical computing is well documented. Graphics cards for certain generic types of computation offer parallel processing speedups with advantages. They are Cost: graphics cards are relatively cheap, being commodity products. Accessibility: graphics cards are readily obtainable from consumer-level computer stores or over the internet. Maintenance: the devices are self-contained and can be hosted on conventional desktop and laptop computers. Speed: in line with multi-core CPU clusters, graphics cards offer significant speedup, albeit for a restricted class of scientific computing algorithms. Power: GPUs are low energy consumption devices compared to clusters of traditional computers, with a graphics card requiring around 200 Watts. While improvements in energy efficiency are application-specific, it is reasonable in many situations to expect a GPU to use around 10 per cent of the energy to that of an equivalent CPU cluster. Dedicated and local: the graphics cards slot into conventional computers offering the user ownership without the need to transport data externally.

The parallelization of the advanced Monte Carlo methods described here opens up challenges both for practitioners and for algorithm designers. There are already an abundance of statistical problems that are being solved computationally and technological advances, if taken advantage of by the community, can serve to make previously impractical solutions eminently reasonable and motivate the development of new methods.

The speedups have practical significance. Arithmetic intensity is important. There is a roughly linear penalty for the space complexity of each thread. Emerging many-core technology is likely to have the same kinds of restrictions. There is a need for methodological attention to this model of computation. For example, SMC sampler methodology can be more suitable to parallelization when the number of auxiliary distributions one wants to introduce is not very large. There are many other algorithms that will benefit from this technology.

Acknowledgements

I would like to express my sincere grateful very much to Professor Dr Soe Mya Mya Aye, Head of Department of Computer Studies, Yangon University for her kind permission to carry out this research. My thanks to my gratitude to Dr. Pho Kaung, Rector, University of Yangon, who has given me invaluable advice and patient guidance that helped my research work to accomplish.

References

1. Andrieu C., Doucet A., and Holenstein R., (2010) "Particle Markov Chain Monte Carlo." *Journal of the Royal Statistical Society, Ser. B*, vol. 72, pp.269–342.

2. Brockwell A. E., (2006) "Parallel Processing in Markov Chain Monte Carlo Simulation by Pre-Fetching." *Journal of Computational and Graphical Statistics*, vol.15,no.1, pp.246–261.

3. Celeux, G., Hurn, M., and Robert, C. P., (2000) "Computational and Inferential Difficulties with Mixture Posterior Distributions." *Journal of the American Statistical Association*, vol.95, pp.957–970.

4. Del Moral P., Doucet, A., and Jasra, A.,(2006) "Sequential Monte Carlo Samplers." *Journal of the Royal Statistical Society, Ser. B*, vol. 68,no.3, pp.411–436.

5. Doucet A., and Johansen A. M.,(2010) *A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later*, in *Handbook of Nonlinear Filtering*, eds., Oxford University Press.