Computations for Markov Chains with Application to Statistical Testing of Systems Based on Usage

Stacy J. Prowell

Preface

This is the second edition of this document giving the basic computations for Markov chain usage models along with their derivations and a method to compute each. There are many motivations for revisiting this document, including providing more detailed derivations, including additional results not available in the prior edition, and providing the algorithms in R. The previous edition provided the algorithms in Scilab, but more and more the author has found that collaborators are more comfortable working with R. The Scilab algorithms do, of course, still work, and are given in Appendix

Acknowledgments

THIS document began life as a summary of Markov computations to answer questions in an email discussion with Tjin Merry. The discussion of Markov chain usage modeling, of the related computations, and of Markov chain-based testing contributed greatly to my understanding of the subject, and I thank Tjin Merry for prompting this document, and for patience with my explanations.

Many people have contributed to this manuscript. I was first introduce to the topic of Markov chain usage models by Drs. Jesse Poore and James Whittaker. Michael Corum provided assistance by reading and using the first version of this document. Special thanks are due Zizhong Chen, who carefully proofread an early draft of the first version and uncovered many careless mistakes. Any mistakes which remain, or which were introduced in this new draft are mine alone.

Contents

List of Figures

List of Tables

CHAPTER 1

Preliminary Definitions

This chapter summarizes some basic results used in the remainder of the document. See [**1**] for a discussion of Matrix analysis and see [**3**] for a discussion of probability theory.

1.1. R

The algorithms presented in this document can be executed using R. This is a freely-available software package which runs on many different platforms. Information about R is available online.

```
https://www.r-project.org/
```

A discussion of R is well outside the scope of this text, but a few matters can be introduced here. First, to create a matrix we specify the data as a vector in column-major form, and then give R one of the two dimensions—either the number of columns or the number of rows.

```
M <- matrix(c(0.1,0.3,0.2,0.4), nrow=2)
```

Note the order of the entries! The above assigns the following matrix to `M`.

$$\begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}$$

The diagonal matrix of size `n` can be obtained via `diag(n)`. Likewise if one has a matrix `M`, one can obtain the diagonal with `diag(M)`. The Hadamard (componentwise) product of two matrices `A` and `B` is obtained via `A*B` and the usual matrix product via `A%*%B`. The inverse of a matrix `M` can be obtained with `solve(M)`.

This should be enough to get the reader started. Further information is available online and in the many excellent print references. R code presented here has been tested with R version 3.4.2, GUI 1.70, running on OS X, and has been formatted with the "format source code" function of the R editor.

1.2. Notation

Throughout this document the term "iff" will be understood to mean "if and only if."

Every *random variable* X has an associated probability distribution $P : X \to [0, 1]$ such that $\int_X dP = 1$. For a random variable X with associated probability distribution P, the probability of a particular outcome $\Pr[X = x]$ will be denoted $p(x)$. Thus if one has two random variables X and Y, and writes $p(x)p(y)$, it should be understood that this denotes $\Pr[X = x] \Pr[Y = y]$.

The *joint probability* $\Pr[X = x \wedge Y = y]$ will be denoted $p(x\&y)$. The conditional probability $\Pr[Y = y \,|\, X = y]$ will be denoted $p(y|x)$.

The *Kronecker delta* $\delta_{i,j}$ extended to the real numbers is defined for $i \in \mathbb{R}$ as follows.

$$\delta_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Both matrices and random variables will be denoted by upper-case letters, and elements of matrices by lower-case letters. For example, the $m \times n$ matrix A consists of mn elements $a_{i,j}$, where i and j are the row and column indices of the element, respectively. To make the association clear, matrix A may be written $A = [a_{i,j}]$. It will often be useful to refer to a single column or row of a matrix. We will do this by writing a star ($\star$) for an index, so that $[a_{i,\star}]$ is the ith row of a matrix, and $[a_{\star,j}]$ is the jth column of a matrix. For any matrix $A = [a_{i,j}]$ we let $A_d = [a_{i,j}\delta_{i,j}]$ denote the matrix consisting of the diagonal of A, with zeros for all off-diagonal entries. Whenever necessary we let $U = [1]$ denote a matrix of ones of appropriate size. The Hadamard product of two matrices $A = [a_{i,j}]$ and $B = [b_{i,j}]$ of the same shape is denoted $A \circ B = B \circ A = [a_{i,j}b_{i,j}]$.

1.3. Expectation of random variables

The *expectation* of a random variable X, denoted $\mathrm{E}[X]$, is an unbiased estimator of the random variable's value. In the discrete case this expectation is a probability-weighted average of the random variable's values.

$$\mathrm{E}[X] = \sum_{x \in X} x \cdot p(x)$$

As a special case consider a random variable C that takes on the value x with probability one. Then $\mathrm{E}[C] = x$. In other words the expectation of a constant is just that constant. Since the value of an expectation is itself a constant, we can conclude that $\mathrm{E}[\mathrm{E}[X]] = \mathrm{E}[X]$.

Mathematical expectation exhibits several useful properties, some of which are presented here without proof.

- *Linearity*: $\mathrm{E}[\sum_{i=1}^{n} a_i X_i] = \sum_{i=1}^{n} a_i \mathrm{E}[X_i]$
- *Positivity*: If $X \geq 0$ then $\mathrm{E}[X] \geq 0$. If $X \geq Y$ then $\mathrm{E}[X] \geq \mathrm{E}[Y]$
- $\mathrm{E}[|X|] \geq |\mathrm{E}[X]|$
- *The Cauchy-Schwarz Inequality*: $|\mathrm{E}[XY]|^2 \leq \mathrm{E}\left[X^2\right] \mathrm{E}\left[Y^2\right]$; equality holds iff there is a real constant λ such that $\lambda X + Y = 0$.
- *Jensen's Inequality*: Let $g : \mathbb{R} \to \mathbb{R}$ be a convex Borel[1] function. Then $g(\mathrm{E}[X]) \leq \mathrm{E}[g(x)]$.

For two random variables X and Y, we conclude the following.

$$\mathrm{E}[XY] = \sum_{x \in X, y \in Y} xy \cdot p(x \& y)$$

[1]Borel sets and functions are important to a rigorous study of probability, but can be ignored for the purpose of this text.

Iff the two random variables are independent, then $p(x\&y) = p(x)p(y)$. In this case we have the following short derivation of an important property.

$$\begin{aligned}
\mathrm{E}[XY] &= \sum_{x\in X, y\in Y} xy \cdot p(x\&y) \\
&= \sum_{x\in X, y\in Y} xy \cdot p(x)p(y) \\
&= \sum_{x\in X, y\in Y} x \cdot p(x) \cdot y \cdot p(y) \\
&= \left(\sum_{x\in X} x \cdot p(x)\right)\left(\sum_{y\in Y} y \cdot p(y)\right) \\
&= \mathrm{E}[X]\,\mathrm{E}[Y] \qquad (1.3.1)
\end{aligned}$$

1.4. Variance and covariance of random variables

How far can one expect the value of a random variable X to differ from its expectation $\mathrm{E}[X]$? That is, what is the expectation of $|X - \mathrm{E}[X]|$? This question is answered by the *variance* $\mathrm{Var}[X]$ of the random variable, which is the expectation of the square of the difference between the observed value and the expected value.

$$\mathrm{Var}[X] = \mathrm{E}\big[(X - \mathrm{E}[X])^2\big]$$

One can "scale" the variance back by taking the square root. This is called the *standard deviation* $\sigma[X] = \sqrt{\mathrm{Var}[X]}$. The advantage of the standard deviation is that it has the same dimensionality and units as the random variable itself.

The variance of a random variable X can be restated using linerarity and the definition of expectation as follows.

$$\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}\big[(X - \mathrm{E}[X])^2\big] \\
&= \mathrm{E}\big[X^2 - 2X\mathrm{E}[X] + \mathrm{E}^2[X]\big] \\
&= \mathrm{E}\big[X^2\big] - \mathrm{E}[2X\mathrm{E}[X]] + \mathrm{E}^2[X] \\
&= \mathrm{E}\big[X^2\big] - 2\mathrm{E}[X]\,\mathrm{E}[X] + \mathrm{E}^2[X] \\
&= \mathrm{E}\big[X^2\big] - \mathrm{E}^2[X] \qquad (1.4.1)
\end{aligned}$$

Eq. 1.4.1 is a commonly-encountered, and useful, formula for variance.

Consider two random variables, X and Y. We already know from eq. 1.3.1 that if X and Y are independent we have $\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y]$. It is reasonable to ask how much $\mathrm{E}[XY]$ and $\mathrm{E}[X]\,\mathrm{E}[Y]$ differ when the two are *not* independent. This is the *covariance* of X and Y, and it is defined as follows.

$$\mathrm{Cov}[X, Y] = \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]$$

Obviously if X and Y are independent, then one has $\mathrm{Cov}[X, Y] = 0$. Consider the special case where X and Y are "completely dependent" ($X = Y$), such as when one state of a Markov chain unconditionally follows another in all realizations. In

this special case one has the following derivation.

$$\begin{aligned}\mathrm{Cov}[X,Y] &= \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y] \qquad\qquad \text{for independent } X, Y\\ &= \mathrm{E}\big[X^2\big] - \mathrm{E}^2[X]\\ &= \mathrm{Var}[X] = \mathrm{Var}[Y]\end{aligned}$$

Consider the variance of the sum of two random variables X and Y. This can be written as follows.

$$\begin{aligned}\mathrm{Var}[X+Y] &= \mathrm{E}\big[(X+Y)^2\big] - \mathrm{E}^2[X+Y]\\ &= \mathrm{E}\big[X^2 + 2XY + Y^2\big] - (\mathrm{E}[X+Y])^2\\ &= \mathrm{E}\big[X^2\big] + 2\mathrm{E}[XY] + \mathrm{E}\big[Y^2\big] - (\mathrm{E}[X] + \mathrm{E}[Y])^2\\ &= \mathrm{E}\big[X^2\big] + 2\mathrm{E}[XY] + \mathrm{E}\big[Y^2\big]\\ &\quad - (\mathrm{E}^2[X] + 2\mathrm{E}[X]\,\mathrm{E}[Y] + \mathrm{E}^2[Y])\\ &= \big(\mathrm{E}\big[X^2\big] - \mathrm{E}^2[X]\big) + \big(\mathrm{E}\big[Y^2\big] - \mathrm{E}^2[Y]\big)\\ &\quad + 2\mathrm{E}[XY] - 2\mathrm{E}[X]\,\mathrm{E}[Y]\\ &= \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\,(\mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y])\\ &= \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,Y]\end{aligned}$$

If X and Y are independent random variables, one has $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$. (The converse is not true; $\mathrm{Cov}[X,Y] = 0$ does *not* guarantee that X and Y are independent.)

An alternate way to think about the covariance is to consider the "joint variance," $\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]$.

$$\begin{aligned}&\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]\\ &\quad= \mathrm{E}[XY - X\mathrm{E}[Y] - Y\mathrm{E}[X] + \mathrm{E}[X]\,\mathrm{E}[Y]]\\ &\quad= \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y] - \mathrm{E}[X]\,\mathrm{E}[Y] + \mathrm{E}[X]\,\mathrm{E}[Y]\\ &\quad= \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]\\ &\quad= \mathrm{Cov}[X,Y]\end{aligned}$$

If X and Y are not independent, then one can use the Cauchy-Schwarz inequality to obtain an upper bound for $\mathrm{Var}[X+Y]$ (if the covariance is not readily computable and no better bound is available).

$$\begin{aligned}|\mathrm{Cov}[X,Y]|^2 &= |\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]|^2\\ &= |\mathrm{E}[UV]|^2\\ &\le \mathrm{E}\big[U^2\big]\,\mathrm{E}\big[V^2\big]\\ &\le \mathrm{E}\big[(X - \mathrm{E}[X])^2\big]\,\mathrm{E}\big[(Y - \mathrm{E}[Y])^2\big]\\ &\le \mathrm{Var}[X]\,\mathrm{Var}[Y] \qquad (1.4.2)\end{aligned}$$

Eq. 1.4.2 bounds the covariance as $|\mathrm{Cov}[X,Y]| \le \sqrt{\mathrm{Var}[X]\,\mathrm{Var}[Y]}$.

While the variance does not have as many nice properties as the expectation, a more intuitive sense of what it means can be obtained by applying the *Chebyshev inequality*. This states that the probability that the observed value x of random

variable X differs from the expectation $\mathrm{E}[X]$ by at least k, for some positive k, is bounded above by $\mathrm{Var}[X]/k^2$.

$$\Pr[|X - \mathrm{E}[X]| \geq k] \leq \frac{\mathrm{Var}[X]}{k^2} \tag{1.4.3}$$

Note that nothing has been said about the distribution of X; eq. 1.4.3 is a *statistically conservative* upper bound. It may be possible to do much better if more is known about the random variable.

The probability that an observation of a random variable falls in a given interval is the *confidence*. As an example, we can apply the Chebychev inequality to compute the 95% confidence interval.

$$\begin{aligned} 0.95 &\leq \frac{\mathrm{Var}[X]}{k^2} \\ k^2 &\leq \frac{\mathrm{Var}[X]}{0.95} \\ k &\leq \sqrt{\frac{\mathrm{Var}[X]}{0.95}} \end{aligned}$$

We interpret this to say that *at least* 95% of the time the observed value of X will be in the following closed interval, which we can refer to as the "95% confidence interval."

$$\left[\mathrm{E}[X] - \sqrt{\frac{\mathrm{Var}[X]}{0.95}}, \mathrm{E}[X] + \sqrt{\frac{\mathrm{Var}[X]}{0.95}}\right]$$

CHAPTER 2

Markov Computations

MANY of the results and derivations in this chapter are taken directly from [**2**], though the derivations presented here are likely much more detailed. The former reference focuses on Markov chains more generally, and is the appropriate source to consult for general questions.

2.1. Representing models

Consider the simple model of fig. 2.1.1. We can express this model in a computer-readable manner in many different ways, but one relatively human-friendly manner is TML. TML is a text-based language explicitly developed for representing Markov chain usage models [**4**]. The given model is equivalent to the following TML.

```
model example
[Enter]  ($  1  $)  "a"  [A]
[A]      ($ 1/2 $)  "b"  [B]
         ($ 1/2 $)  "c"  [C]
[B]      ($ 1/2 $)  "b"  [B]
         ($ 1/4 $)  "c"  [C]
         ($ 1/4 $)  "e"  [Exit]
[C]      ($ 1/4 $)  "a"  [A]
         ($ 1/2 $)  "e"  [Exit]
         ($ 1/4 $)  "f"  [Exit]
end
```

A short primer on TML is included as Appendix A, but we note that whitespace is not significant to TML, and the model could have been written in the following form, where the probabilities are inferred.

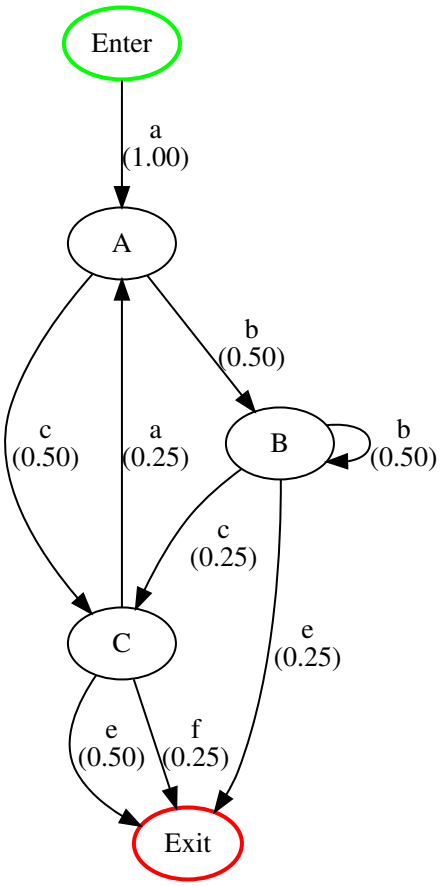

FIGURE 2.1.1. Example model

```
($assume(1) normalize(1)$)
model example
[Enter]   "a" [A]
[A]       "b" [B]
          "c" [C]
[B] ($2$) "b" [B]
          "c" [C]
          "e" [Exit]
[C]       "a" [A]
    ($2$) "e" [Exit]
          "f" [Exit]
end
```

There are two special states in this model: the *source* or start state `[Enter]`, and the *sink* or stop state `[Exit]`. The sink state represents the end of a single use and thus no usage events are possible from this state.

This model can be described by two matrices. For these matrices let the states be indexed in the order `[Enter]`, `[A]`, `[B]`, `[C]`, and `[Exit]`, and let the stimuli be indexed in alphabetic order. The first matrix is the *(state) transition matrix*, $P = [p_{i,j}]$, for which $p_{i,j}$ is the probability that the next state is j given that the current state is i. For the example model the transition matrix is the following.

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 & 0 & \frac{3}{4} \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note that the example matrix has been made *recurrent* by connecting `[Exit]` back to `[Enter]` with probability one. This will be important for several computations, and is assumed where necessary. Note that there are two transitions from state `[C]` to state `[Exit]`; these two transitions are mutually exclusive, so the probability that the next state is `[Exit]` given that the current state is `[C]` is the sum of the two: $\frac{1}{4} + \frac{1}{2} = \frac{3}{4}$. We can represent the transition matrix in R as follows, remembering that we need to use column-major form.

```
P <- matrix(c(
  0,   0,   0,   0,   1,
  1,   0,   0, 1/4,   0,
  0, 1/2, 1/2,   0,   0,
  0, 1/2, 1/4,   0,   0,
  0,   0, 1/4, 3/4,   0), nrow=5)
```

For many computations it will be useful to have a reduced matrix $Q = [q_{i,j}]$ in which the row and column for the model sink have been removed. For the example model the reduced matrix is the following.

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 & 0 \end{bmatrix}$$

Both P and Q are always square matrices. We can extract the Q matrix from the P matrix in R as follows.

```
n <- nrow(P)
Q <- P[-n,-n]
```

The second matrix is the *stimulus (occurrence) matrix* $S = [s_{i,j}]$, for which $s_{i,j}$ is the probability that the next usage event (or *stimulus*) will be j, given that the current state is i. For the example model the stimulus matrix is the following.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

The usage models used here are deterministic; a stimulus can only label one outgoing transition from a state, so there is no need to sum transition probabilities in the stimulus matrix. We can represent the stimulus matrix in R as follows.

```
S <- matrix(c(
  1,   0,   0, 1/4,
```

```
0, 1/2, 1/2,   0,
0, 1/2, 1/4,   0,
0,   0, 1/4, 1/2,
0,   0,   0, 1/4), nrow=4)
```

It will occasionally be convenient to discuss the probability of a transition from state i to state j labeled with stimulus k. This probability will be denoted $p_{i,j,k}$ when needed. The *restriction* of a matrix A to just those elements corresponding to stimulus k will be denoted $A|_k$. Thus $Q|_k = [q_{i,j,k}]$ for fixed k, and we also have the following.

$$Q = \sum_k Q|_k$$

Throughout the rest of this document, the source will be assumed to have index one, while the sink will be assumed to have the highest index n. The number of stimuli for a model will be denoted s. Thus P is always an $n \times n$ matrix, while S is always an $(n-1) \times s$ matrix, since there can be no outgoing arcs (other than the unlabeled recurrence loop) from the sink.

For every state i in a connected model, there is a non-zero probability path from the source to i, and a non-zero probability path from i to the sink. Because of the recurrence loop the graph of the model is strongly connected, meaning that from any state i there is a path to any other state j. It follows that P is an irreducible matrix (see Theorem 6.2.24 in [**1**]). All models will be assumed to be connected in this manner.

2.2. Computing the number of occurrences of a state in a test case

The number of occurrences of a state in a random walk from source to sink (a test case) can be computed using the reduced matrix Q. If the sink is made absorbing (letting $p_{n,n} = 1$ and $p_{n,1} = 0$), then one can compute the number of occurrences of a state prior to absorption at the sink. Let T denote the set of transient (non-absorbing) states, which will be every state other than the sink. Let $n_{i,j}$ be a random variable counting the number of occurrences of state j prior to absorption, given that one starts in state i (if $i = j$ then we count the initial occurence, too, and use $\delta_{i,j}$ to do that). From i we may move to absorbing state k (in our models, the sink n) with probability $p_{i,k}$ and gain contribution $\delta_{i,j}$. Otherwise we move to a transient state k and gain contribution $\delta_{i,j}$ from the initial state plus $n_{k,j}$ from the later steps of the realization. Since these different possibilities are

disjoint, we can simply sum them up.[1]

$$\begin{aligned}
\mathrm{E}[n_{i,j}] &= \mathrm{E}\left[\sum_{k\notin T} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}(\delta_{i,j} + n_{k,j})\right] \\
&= \mathrm{E}\left[\sum_{k\notin T} p_{i,k}\delta_{i,j}\right] + \mathrm{E}\left[\sum_{k\in T} p_{i,k}(\delta_{i,j} + n_{k,j})\right] \\
&= \sum_{k\notin T} \mathrm{E}[p_{i,k}\delta_{i,k}] + \sum_{k\in T} \mathrm{E}[p_{i,k}(\delta_{i,j} + n_{k,j})] \\
&= \sum_{k\notin T} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}\mathrm{E}[\delta_{i,j} + n_{k,j}] \\
&= \sum_{k\notin T} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}\left(\delta_{i,j} + \mathrm{E}[n_{k,j}]\right) \\
&= \sum_{k\notin T} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}\mathrm{E}[n_{k,j}] \\
&= \sum_{k} p_{i,k}\delta_{i,j} + \sum_{k\in T} p_{i,k}\mathrm{E}[n_{k,j}] \\
&= \delta_{i,j}\sum_{k} p_{i,k} + \sum_{k\in T} p_{i,k}\mathrm{E}[n_{k,j}] \\
&= \delta_{i,j} + \sum_{k\in T} p_{i,k}\mathrm{E}[n_{k,j}] \qquad (2.2.1)
\end{aligned}$$

Let $N = [\mathrm{E}[n_{i,j}]]$ be the matrix of these expectations. Then eq. 2.2.1 can be re-written in matrix form as follows.

$$\begin{aligned}
N &= I + QN \\
N - QN &= I \\
(I - Q)N &= I \\
N &= (I - Q)^{-1} \qquad (2.2.2)
\end{aligned}$$

The matrix N defined by eq. 2.2.2 is called the *fundamental matrix* for absorbing chains. Many Markov chain results can be obtained from this fundamental matrix. The expected number of occurences of each state exists and is finite (because absorption is certain); we can conclude that the inverse in eq. 2.2.2 exists.

What about the variance associated with this expectation? We need to find the second moment $\mathrm{E}\left[n_{i,j}^2\right]$, and we can do this using the same method as we did for $\mathrm{E}[n_{i,j}]$, above. If we start in state i then we may move to an absorbing state k (in the specific case of our models, only the sink n) with probability $p_{i,k}$ and gain contribution $\delta_{i,j}^2$. Otherwise we move to a transient state k and gain the square of the sum of $\delta_{i,j}$ (from the initial state) and $n_{k,j}$ (from the later steps). This gives

[1] This derivation may seem excessive, but we include this detail to provide a pattern for future derivations which may be less obvious. We will skip the initial steps, which are "obvious," in future derivations.

the following equation where, we note, $\delta_{i,j}^2 = \delta_{i,j}$.

$$\begin{aligned}
\mathrm{E}\big[n_{i,j}^2\big] &= \sum_{k \notin T} p_{i,k}\delta_{i,j}^2 + \sum_{k \in T} p_{i,k}\mathrm{E}\big[(\delta_{i,j} + n_{k,j})^2\big] \\
&= \sum_{k \notin T} p_{i,k}\delta_{i,j}^2 + \sum_{k \in T} p_{i,k}\mathrm{E}\big[\delta_{i,j}^2 + 2\delta_{i,j}n_{k,j} + n_{k,j}^2\big] \\
&= \sum_{k \notin T} p_{i,k}\delta_{i,j}^2 + \sum_{k \in T} p_{i,k}\mathrm{E}\big[\delta_{i,j}^2\big] + \mathrm{E}[2\delta_{i,j}n_{k,j}] + \mathrm{E}\big[n_{k,j}^2\big] \\
&= \sum_{k \notin T} p_{i,k}\delta_{i,j} + \sum_{k \in T} p_{i,k}\left(\delta_{i,j} + 2\delta_{i,j}\mathrm{E}[n_{k,j}] + \mathrm{E}\big[n_{k,j}^2\big]\right) \\
&= \sum_{k \notin T} p_{i,k}\delta_{i,j} + \sum_{k \in T} p_{i,k}\delta_{i,j} + \sum_{k \in T} p_{i,k}2\delta_{i,j}\mathrm{E}[n_{k,j}] + \sum_{k \in T} p_{i,k}\mathrm{E}\big[n_{k,j}^2\big] \\
&= \delta_{i,j}\sum_{k} p_{i,k} + 2\delta_{i,j}\sum_{k \in T} p_{i,k}\mathrm{E}[n_{k,j}] + \sum_{k \in T} p_{i,k}\mathrm{E}\big[n_{k,j}^2\big] \\
&= 1 + 2\sum_{k \in T} p_{j,k}\mathrm{E}[n_{k,j}] + \sum_{k \in T} p_{i,k}\mathrm{E}\big[n_{k,j}^2\big] \qquad (2.2.3)
\end{aligned}$$

Let $N_2 = \big[\mathrm{E}\big[n_{i,j}^2\big]\big]$ be the matrix of these expectations. Then eq. 2.2.3 can be re-written in matrix form as follows.

$$\begin{aligned}
N_2 &= I + 2(QN)_d + QN_2 \\
N_2 - QN_2 &= I + 2(QN)_d \\
N_2(I - Q) &= I + 2(QN)_d \\
N_2 &= (I - Q)^{-1}(I + 2(QN)_d) \\
N_2 &= N(I + 2(QN)_d) \\
N_2 &= N + 2N(QN)_d \qquad (2.2.4)
\end{aligned}$$

Since we already know $N = I + QN$, we can conclude that $QN = N - I$. We substitute in eq. 2.2.4 and obtain the following.

$$\begin{aligned}
N_2 &= N + 2N(N - I)_d \\
&= N + 2NN_d - 2N \\
&= 2NN_d - N \\
&= N(2N_d - I) \qquad (2.2.5)
\end{aligned}$$

Now, with eq. 2.2.5 in hand, we can compute the matrix of variances as $[\mathrm{Var}[n_{i,j}]] = N_2 - N \circ N$.

These results allow the computation of the expected number of occurrences for each state and the associated variance by the R function in algorithm 1. Note that for a usage model one is primarily concerned with only the first row of the returned matrices, since one is concerned with the behavior when the model is started from the source. The results obtained for the example model are presented in table 1.

Let l denote the number of state transitions from the source to the sink in a test case. The expected length $\mathrm{E}[l]$ of a test case generated from a Markov chain usage model can be quickly computed once N is known by just summing the expected

Algorithm 1 Compute the expected number of occurrences of each state

```
# Compute the non-terminal expectation and
# variance matrices for the stochastic matrix
# P.
#
# P: a row-stochastic matrix
# return: list(N,V) where
#   N: the expected occurrence of each state
#   V: the associated variances
get_nte <- function(P) {
   n <- nrow(P)
   Q <- P[-n, -n]
   N <- solve(diag(n - 1) - Q)
   V <- N %*% (2 * diag(diag(N)) - diag(n - 1)) - N * N
   list(N, V)
}
```

TABLE 1. Expected occurrence and associated variance for each state

State	Occurrence (visits per test case)	Variance
[Enter]	1.000	0.000
[A]	1.231	0.284 0
[B]	1.231	2.556
[C]	0.923 1	0.497 0
[Exit]	1.000	0.000

occurrences of each state.

$$\mathrm{E}[l] = \mathrm{E}\left[\sum_k n_{1,k}\right] = \sum_k \mathrm{E}[n_{1,k}]$$

That is, the average length of a test case is the number of state transitions from the source to the sink. Note that $\mathrm{E}[l]$ does not include the sink; the total average number of *state visits* (including the sink) is one more, since the sink is always visited exactly once. In a later section we will revisit this and also discuss how to obtain an associated variance.

2.3. Computing the long-run state probabilities

Assume that many test cases (random walks from source to sink) are generated from a usage model. For each state one can sum the number of occurrences of that state, and then divide by this by the total number of occurrences of all states. As the number of test cases becomes very large, this ratio will approach a fixed value called the *long-run occupancy* or *long-run probability* of the state. Since the usage model is always in one of its states, the sum of all long-run occupancies is one.

Let $\Pi = [\pi_1, \pi_2, \ldots, \pi_n]$ be the vector of long-run probabilities for the n states. This vector can be found for a given transition matrix P as the unique stochastic

TABLE 2. Long-run occupancies

State	Long-run Occupancy
`[Enter]`	0.185 7
`[A]`	0.228 6
`[B]`	0.228 6
`[C]`	0.171 4
`[Exit]`	0.195 7

vector solution to the left-eigenvector equation for eigenvalue 1.

$$\Pi = \Pi P \tag{2.3.1}$$

The left-eigenvector Π is sometimes called the *Perron* eigenvector. Eq. 2.3.1 is equivalent to the following system of equations.

$$\begin{aligned}
\pi_1 &= \pi_1 p_{1,1} + \pi_2 p_{2,1} + \cdots + \pi_n p_{n,1} \\
\pi_2 &= \pi_1 p_{1,2} + \pi_2 p_{2,2} + \cdots + \pi_n p_{n,2} \\
&\vdots \\
\pi_n &= \pi_1 p_{1,n} + \pi_2 p_{2,n} + \cdots + \pi_n p_{n,n} \\
1 &= \pi_1 + \pi_2 + \cdots + \pi_n
\end{aligned}$$

Note that there are n unknowns and $n+1$ equations. Incidentally, each row of $P^\infty = \lim_{n\to\infty} P^n$ is equal to the vector Π, unless P is periodic [**2**]. There are many ways to solve this system. For a simple solution by hand, back-substitution will work just fine using the above equations.

The fundamental matrix N can also be used to obtain Π. The average number of state visits in a test case is $\mathrm{E}[l]+1$. The fraction of time one spends in state i in the long run is π_i, so the average number of times state i will be visited in a single test case is $\pi_i(\mathrm{E}[l]+1)$. We conclude the following.

$$\begin{aligned}
\mathrm{E}[n_{1,i}] &= \pi_i \left(\mathrm{E}[l]+1\right) \\
\pi_i &= \frac{\mathrm{E}[n_{1,i}]}{\mathrm{E}[l]+1}
\end{aligned} \tag{2.3.2}$$

Note that $\mathrm{E}[n_{1,n}]$ is not computed by equation 2.2.2, but we can conclude that $\mathrm{E}[n_{1,n}] = 1$ since the sink is always visited exactly once per test case. Algorithm 2 uses this relationship to compute Π. The results for the example model are presented in table 2. The algorithm does not give a value for `[Exit]`, but that can be obtained by subtracting the sum of the other values from one.

An alternate method for obtaining the long run distribution is to use an iterative method called the *power method*. Given the ith guess at the long-run distribution Π_i, one computes a slightly better guess by $\Pi_{i+1} = \Pi_i P$. This method is guaranteed to converge to the Perron eigenvector if the matrix is *primitive* [**1**, p. 516]. Without going into too much detail, if there is a non-zero entry on the diagonal then the matrix is primitive (though the converse is not true). Thus if $p_{j,j} \neq 0$ for any j, then the power method will converge to the Perron eigenvector.

One way to guarantee primitivity is to introduce a "dummy" state d with a self-loop to ensure primitivity (that is, $p_{d,d} > 0$), apply the power method, then "remove" the dummy state and correct for its presence. This can be done by adding

Algorithm 2 Compute the Perron eigenvector (long-run probabilities)

```
# Compute the Perron eigenvector for the
# stochastic matrix P and return it. The
# computation is performed by computation
# of the fundamental matrix.
#
# P: a square row-stochastic matrix
# N: the fundamental matrix, if available
# return: the Perron eigenvector
get_pi <- function(P, N) {
   n <- nrow(P)
   if (nargs() < 2) {
      N <- get_nte(P)[[1]]
   }
   len <- 1 + sum(N[1, ])
   pi <- N[1, ]/len
   append(pi, c(1 - sum(pi)))
}
```

the dummy state on the recurrence loop from the sink to the source, so $p_{n,d} = 1$, $p_{d,d} = 1/2$, and $p_{d,1} = 1/2$. Flow always passes straight through the dummy state, so any of the probability mass absorbed by the dummy state can be evenly redistributed among the other states. This change can be made to the example model's transition matrix P as follows.

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{3}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

(Note that the matrix was primitive *before* this change, since $p_{3,3} \neq 0$.) This change guarantees primitivity by making $p_{d,d} \neq 0$. The power method is then applied to yield the long-run probabilities Π. The entry for the dummy state is dropped from Π and the vector is re-normalized to obtain the correct long-run probabilities. This is actually easier than it sounds; algorithm 3 implements this approach. For the example model convergence to within 1×10^{-6} takes 46 iterations starting from the default guess.

2.4. Performing sensitivity analysis

A change in a transition probability impacts the occupancies of all states since the model is ergodic. Consider the example model and the exit arcs from state `[C]`. If one changes the probabilities of `"e"` or `"f"` to 0.9, this has a significant effect on the model. Alternately, changing the probability of `"a"` to 0.9 has a very different effect. Suppose we are attempting to validate a model against our expectations for what a "use" should look like. We discover that the model is spending far too much time in one state versus another. One way to determine how to adjust model probabilities to correct this is to use sensitivity analysis.

Algorithm 3 Compute the approximation to the Perron eigenvector via the power method

```
# Compute the Perron eigenvector for the row
# stochastic matrix P and return it.  The
# computation is performed using the power
# method.
#
# P: a row-stochastic matrix
# g: (optional) an initial guess
# return: list(y,step) where
#   y: an approximation of the Perron eigenvector
#   step: the number of iterations required
get_pi_approx <- function(P, g) {
   n <- nrow(P)
   d <- n + 1
   P <- rbind(cbind(P, 0), 0)
   P[n, d] = 1
   P[d, d] = 1/2
   P[d, 1] = 1/2
   P[n, 1] = 0
   if (missing(g)) {
      yold <- rep(1/d, d)
   } else {
      g <- c(g, g[n] * 2)
      yold <- g/sum(g)
   }
   y <- yold %*% P
   step <- 1
   limit <- 1e-06
   while (sum(abs(yold - y)) > limit) {
      step = step + 1
      yold = y
      y = y %*% P
   }
   y = y[1:5]
   y = y/sum(y)
   list(y, step)
}
```

For each pair of state i and j in the model such that $0 < p_{i,j} < 1$, we can set $p_{i,j}$ to some value $0 < q < 1$ (adjusting the other probabilities, of course) and compute the long-run occupancies. We will use the following notation for this new vector.

$$\Pi^{(p_{i,j}=q)} = \left[\pi_k^{(p_{i,j}=q)}\right]$$

The *sensitivity* of state k with respect to the transition from state i to state j is defined as follows.

$$z_{i,j,k} = \frac{\pi_k^{(p_{i,j}=0.95)} - \pi_k^{(p_{i,j}=0.05)}}{0.90}$$

TABLE 3. Transition sensitivities

From	To	[Enter]	[A]	[B]	[C]	[Exit]
[A]	[B]	-0.047 25	-0.094 49	0.378 0	-0.189 0	-0.047 24
[A]	[C]	0.047 25	0.094 49	-0.378 0	0.189 0	0.047 24
[B]	[B]	-0.163 9	-0.201 7	0.680 8	-0.151 3	-0.163 9
[B]	[C]	-0.003 152	-0.037 82	-0.182 8	0.138 7	-0.003 151
[B]	[Exit]	0.080 67	0.064 54	-0.161 3	-0.064 54	0.080 67
[C]	[A]	-0.132 7	0.096 50	0.096 50	0.072 37	-0.132 7
[C]	[Exit]	0.132 7	-0.096 50	-0.096 50	-0.072 37	0.132 7

The matrix of sensitivities is computed by algorithm 4. The sensitivities for the example model are given in table 3. We interpret the sensitivities as the effect that increasing the transition probability has on the long-run occupancy of each state. That is, we would expect that increasing the transition probability from state [A] to state [B] will decrease the long-run occupancy of state [C] and increase the long-run occupancy of state [B].

Suppose that one wishes to direct more of the model's flow to state [C]. Table 3 indicates that the best way to do so is to increase the probability associated with the arc from state [A] to state [C], as this is where we find the largest positive value in the column for state [C]. The best way to increase the length of a test case is to decrease the long-run occupancy of state [Exit], which can be done by increasing the probability associated with the transition from state [B] to state [B].

2.5. Computing other long run statistics

The long run occupancy of the transition from state i to state j is obtained as $\pi_i p_{i,j}$. The long run occupancy for a particular stimulus σ_k can likewise be obtained by summing the long run occupancies of the arcs labeled with the stimulus.

$$\sigma_k = \frac{1}{1-\pi_n} \sum_{i=1}^{n-1} \pi_i s_{i,k}$$

Algorithm 4 Compute the matrix of sensitivities

```
# Compute the matrix of arc sensitivities.
# The first column of the returned matrix
# is the source state, the second column is
# the target state. The remaining columns
# are the changes in the occupancies.
#
# P: the state transition matrix
# pi: the (optional) long run occupancies
# return: the sensitivities matrix
get_sensitivities <- function(P, pi) {
   n <- nrow(P)
   if (missing(pi)) {
      pi <- get_pi_approx(P)
   }
   Z <- matrix(rep(c(0), (n + 2)), nrow = 1)
   m <- 1
   for (i in 1:n) {
      for (j in 1:n) {
         if (0 < P[i, j] && P[i, j] < 1) {
            t <- P[i, ]
            x <- 1 - P[i, j]
            P[i, ] = t/x * 0.05
            P[i, j] = 0.95
            ph <- get_pi_approx(P, pi)[[1]]
            P[i, ] = t/x * 0.95
            P[i, j] = 0.05
            pl <- get_pi_approx(P, pi)[[1]]
            Z = rbind(Z, 0)
            Z[m, 1:2] = c(i, j)
            Z[m, 3:(n + 2)] = (ph - pl)/0.9
            m = m + 1
            P[i, ] = t
         }
      }
   }
   Z[1:m - 1, ]
}
```

Bibliography

[1] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, England, 1985.

[2] J.G. Kemmeny and J.L. Snell. *Finite Markov Chains*. Springer-Verlag, New York, NY, 1976.

[3] P.E. Pfeiffer. *Concepts of Probability Theory*. second revised edition. Dover Publications, Inc., New York, NY, 1978.

[4] S.J. Prowell. TML: A description language for markov chain usage models. *Information and Software Technology*, 42(12):835–844, September 2000.

APPENDIX A

A Brief TML Primer

APPENDIX B

Algorithms for Scilab

The algorithms presented in this appendix can be executed using Scilab, a freely-available software package that runs on many different platforms. While the Scilab input language is very similar to MATLAB's input language, there are differences and the algorithms presented here will require modification to run under MATLAB. Information about Scilab is available from Inria: `https://www.scilab.org`.

Algorithm 5 Compute the expected number of occurrences of each state

```
// Compute the non-terminal expectation and
// variance matrices for the stochastic matrix
// P.
//
// P: a row-stochastic matrix
// N: the expected occurrence of each state
// V: the associated variances
function [N,V]=get_nte(P),
  n=size(P,1)-1;
  Q=P(1:n,1:n);
  N=inv(eye(n,n)-Q);
  V=N*(2*diag(diag(N))-eye(n,n))-(N.*N);
endfunction;
```

Algorithm 6 Compute the Perron eigenvector (long-run probabilities)

```
// Compute the Perron eigenvector for the
// stochastic matrix P and return it. The
// computation is performed by computation
// of the fundamental matrix.
//
// pi: the Perron eigenvector
// P: a square row-stochastic matrix
// N: the fundamental matrix, if available
function [pi]=get_pi(P,N),
  n=size(P,1);
  if argn(2)<2 then
    [N,V]=get_nte(P);
  end;
  len=1.0;
  for i=1:n-1, len=len+N(1,i); end;
  for i=1:n-1, pi(1,i)=N(1,i)/len; end;
  pi(1,n)=1/len;
endfunction;
```

APPENDIX C

The Normal Distribution

Here we consider the normal, or Gaussian, distribution more closely, starting with a definition of just what it means to be "normally" distributed.

DEFINITION 1. Consider a random variable X, and let the frequency with which the value x is observed be $f(x)$. Then the random variable X is *normally distributed* iff the rate at which the frequency $f(x)$ decreases is proportional to both the distance of x from the mean $E[X] = \mu$ and the frequency $f(x)$ itself. Thus $df(x)/dx \propto (x - \mu)f(x)$.

Let the proportionality constant be some $k \geq 0$. Then we can write the following.

$$\begin{aligned} \frac{df(x)}{dx} &= -k(x-\mu)f(x) \\ df(x) &= -k(x-\mu)f(x)dx \\ \frac{df(x)}{f(x)} &= -k(x-\mu)dx \end{aligned}$$

We can immediately integrate[1] both sides as follows.

$$\begin{aligned} \int \frac{1}{f(x)} df(x) &= \int -k(x-\mu)dx \\ \ln f(x) &= -k\frac{(x-\mu)^2}{2} + \ln C \end{aligned}$$

Note that we have chosen to write $\ln C$ instead of C. Now we take the exponential of both sides to eliminate the logarithm on the left.

$$\begin{aligned} \exp(\ln f(x)) &= \exp\left[-k\frac{(x-\mu)^2}{2} + \ln C\right] \\ f(x) &= Ce^{-\frac{k}{2}(x-\mu)^2} \end{aligned}$$

At this point we have two unknown constants: C and k. Everything up to this point has been rather simple, but finding k and C turns out to be a bit more complicated. We will need to use *substitution* and to convert to *polar coordinates*. Neither is really that hard.

---

[1]The easy way to do this is to use the substitution $u = x - \mu$, and thus $du = dx$. If you instead used the summation rule and ended up with $-k(x^2/2 - \mu x) + \ln D$ you might be demanding to know which answer is right. These answers are the same; they only differ by the constant. To see this, differentiate both with respect to x, or just multiply out $(x-\mu)^2/2$. One obtains $x^2/2 - \mu x + \mu^2/2$. But $\mu^2/2$ is itself just a constant that can be subsumed into $\ln D$. Alternately we can replace $\ln D$ with $\mu^2/2 + \ln C$.

Note that the area under the curve must be equal to one, since some outcome is certain.

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} f(x)dx \\ &= \int_{-\infty}^{\infty} Ce^{-\frac{k}{2}(x-\mu)^2}dx \\ &= C\int_{-\infty}^{\infty} e^{-\frac{k}{2}(x-\mu)^2}dx \end{aligned} \tag{C.0.1}$$

Now we can substitute. Let u be defined as follows.

$$\begin{aligned} u^2 &= \frac{k}{2}(x-\mu)^2 \\ u &= \sqrt{\frac{k}{2}}(x-\mu) \end{aligned}$$

We differentiate u with respect to x.

$$\begin{aligned} \frac{du}{dx} &= \frac{d}{dx}\sqrt{\frac{k}{2}}(x-\mu) \\ \frac{du}{dx} &= \sqrt{\frac{k}{2}}\frac{d}{dx}(x-\mu) \\ \frac{du}{dx} &= \sqrt{\frac{k}{2}} \\ du &= \sqrt{\frac{k}{2}}dx \end{aligned} \tag{C.0.2}$$

We can thus conclude that $dx = \sqrt{\frac{2}{k}}du$.

Substituting equation equation (C.0.2) into equation equation (C.0.1) we obtain the following.

$$\begin{aligned} 1 &= C\int_{-\infty}^{\infty} e^{-\frac{k}{2}(x-\mu)^2}dx \\ &= C\int_{-\infty}^{\infty} e^{-u^2}\sqrt{\frac{2}{k}}du \\ &= C\sqrt{\frac{2}{k}}\int_{-\infty}^{\infty} e^{-u^2}du \end{aligned}$$

This integral is tricky. While any good table of integrals should contain a solution, let's figure it out. We can square the integral.[2]

$$\begin{aligned} 1 &= \left[C\sqrt{\frac{2}{k}}\int_{-\infty}^{\infty} e^{-u^2}du\right]^2 \\ &= \frac{2C^2}{k}\left[\int_{-\infty}^{\infty} e^{-u^2}du\right]\left[\int_{-\infty}^{\infty} e^{-v^2}dv\right] \\ &= \frac{2C^2}{k}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[e^{-u^2}du\right]\left[e^{-v^2}dv\right] \\ &= \frac{2C^2}{k}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(u^2+v^2)}du\,dv \end{aligned} \tag{C.0.3}$$

Note that we have to use different variables for the two integrals; otherwise we might run into problems. While it seems we are just making a mess, note that we have $u^2 + v^2$. This is the equation[3] for the square of the radius (r^2) of a circle, so let's convert to polar coordinates and see what we get. Now the integral will run from 0 to 2π for θ (the angle) and from 0 to ∞ for the radius (r). In this case we have $r^2 = u^2 + v^2$, $u = r\cos\theta$, and $v = r\sin\theta$. We employ a change of variable, compute the Jacobian[4], and obtain $du\,dv = r\,dr\,d\theta$. Rewriting equation equation (C.0.3) in polar coordinates, we obtain the following.

$$\begin{aligned} 1 &= \frac{2C^2}{k}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(u^2+v^2)}du\,dv \\ &= \frac{2C^2}{k}\int_{0}^{2\pi}\int_{0}^{\infty} re^{-r^2}dr\,d\theta \end{aligned} \tag{C.0.4}$$

---

[2]The derivation looks quite straightforward, but we are actually ignoring a complication that can arise when the limits of an integral are not finite. In this case it does not matter.

[3]It would be more obvious if we had used x and y, but we did not want to confuse the reader, since x had been used previously.

[4]We compute the Jacobian as follows.

$$\begin{aligned} \frac{\partial(u,v)}{\partial(r,\theta)} &= \begin{vmatrix} \frac{\partial u}{\partial r} & \frac{\partial u}{\partial \theta} \\ \frac{\partial v}{\partial r} & \frac{\partial v}{\partial \theta} \end{vmatrix} \\ &= \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} \\ &= r\cos^2\theta - (-r)\sin^2\theta \\ &= r(\cos^2\theta + \sin^2\theta) \\ &= r \end{aligned}$$

From this we obtain the area differential $dA = |r|dr\,d\theta = r\,dr\,d\theta$, since $r$is non-negative.

Equation equation (C.0.4) is much better. Note that if we let $w = -r^2$ then we have $dw = -2r\,dr$. Making this substitution we obtain the following.

$$\begin{aligned} 1 &= \frac{2C^2}{k}\int_0^{2\pi}\int_0^{\infty} e^{-r^2}\left(\frac{1}{2}\right)(2r)\,dr\,d\theta \\ &= -\frac{C^2}{k}\int_0^{2\pi}\int_0^{-\infty} e^{w}\,dw\,d\theta \\ &= -\frac{C^2}{k}\int_0^{2\pi}[e^{w}]_0^{-\infty}\,d\theta \\ &= -\frac{C^2}{k}\int_0^{2\pi}(0-1)\,d\theta \\ &= -\frac{C^2}{k}\int_0^{2\pi}(-1)\,d\theta \\ &= -\frac{C^2}{k}[\theta]_0^{2\pi} \\ &= -\frac{C^2}{k}(2\pi - 0) \\ &= -\frac{2\pi C^2}{k} \end{aligned} \tag{C.0.5}$$

From equation equation (C.0.5) we conclude that $C = \sqrt{k/(2\pi)}$.

Index