NEW DIMENSIONS P1112 EURESCOM project

Computing TCP equilibrium sensitivity with Markov chains

BT31_01.pdf

Contribution to PIR 3.1

Author:

Keith Briggs

EDIN 0001-xxxx

Project P1112

EURESCOM confidential

June 2002

Computing TCP equilibrium sensitivity with Markov chains

Keith Briggs
BTexact Technologies
Adastral Park Antares 2pp5
Martlesham Heath, Suffolk IP5 3RE, UK
Keith.Briggs@bt.com

2002 June 14 12:00

Abstract

I give some background to the TCP congestion-control protocol, followed by a short introduction to Markov chains, and show how they are suited to analyzing TCP. I apply this to the problem of computing the distribution of TCP window size, and in particular to computing the sensitivity of the distribution to the packet loss probability. The novel feature of this approach is to compute the exact functional form of the invariant distribution as a function of packet loss probability.

1 Introduction

The ultimate aims of this work are:

- to build accurate models of various TCP algorithms;
- to analyze them and compare mean behaviour such as throughput;
- to study short-term behaviour (transient response);
- to generalize to non-Markovian dynamics;
- to design new TCPs for multicast, wireless etc.; and
- to design software tool for easy analysis of modified algorithms.

The present paper is a progress report for this long-term project and has the following outline:

- TCP background and terminology
- Markov chain basics
- Mini-TCP toy model
- Markov chain perturbation theory
- Results for mini-TCP
- Results for NewReno TCP

2 TCP background and terminology

A TCP segment or packet consists of a 20 byte header and up to 64K bytes of data in the format shown in figure 1. By a *window* is meant a single continuous burst of segments. A *timeout* in receiving an ACK is taken to indicate a lost packet, which must be retransmitted. The *congestion window* is the transmitter's current estimate of maximum window size to avoid congestion. The *threshold* is the crossover point between slow-start (exponential) phase and congestion-avoidance phases. By *goodput* we mean the number of packets successfully transmitted *and* received per unit time.

source	destination
sequence number	
acknowledgement number	
flags	window size
checksum	urgent
options	
data	

Figure 1: TCP segment header

TCP is intended to provide packet transmission rate control and reliable transmission for application level software, and thus to interface to the underlying IP layer. It is full duplex and adjusts its send rate to adapt to network congestion and receiver buffer capacity. A TCP transmitter must do this with the limited information contained in the acknowledgement (ACK) signals from the receiver.

TCP started with the BSD implementation in the early 1980s. It came to maturity with Jacobson's congestion control work [Jacobson & Karels 1988], and has since developed into numerous varieties: (old)Tahoe, (new)Reno, Westwood, MSWin, linux versions derived from BSD, etc. While these varieties have many sophisticated features, in essence they are all variations on the theme of making a linear increase in window size after a successful transmission, and an geometric decrease after a packet loss. It is rumoured that some implementations have become so complex that one cannot be certain exactly what algorithm they are performing, or that they are bug-free.

The *environment* in which TCP works is the whole portion of the internet on the path to the receiver; but the transmitter can have no knowledge of the traffic density on these links. It is only aware that some packets are not acknowledged before timeout, and so must be assumed to be lost. Thus, the mathematical model of this situation contains only a single parameter p, the packet loss probability. In reality, successive loss events are not independent, and so a strict Markov model is not correct. However, to make the analysis tractable, we must assume the Markov property.

3 Markov chains

We deal with evolving states $x_k \in \mathbb{R}^n$ in discrete time $k = 0, 1, 2, \ldots$. The dimension n of the state space is typically large (hundreds or thousands) in TCP applications. Let P be a matrix of transition probabilities: $P_{ij} = \Pr[x_{k+1} = j \mid x_k = i]$. Letting $\mathbf{1} = [1, 1, 1, \ldots]^T$ be a vector of ones, we have $P > 0$ and $P\mathbf{1} = \mathbf{1}$, so that P is a positive definite, stochastic matrix. We let $Q = I - P$ so that $Q\mathbf{1} = 0$.

Under these conditions the Markov chain has a invariant distribution z which satisfies

$$z^T Q = 0 \qquad \Rightarrow \qquad z^T P = z^T.$$

It may be normalized by $\mathbf{1}^T z = 1$. I will make much use of the fundamental matrix, defined by $F = Q^{\#} = (Q - \mathbf{1}z^T)^{-1}$.

To compute the invariant distribution z we may either:

- find the one-dimensional kernel (rank(Q)=$n - 1$, for example by LU decomposition) of Q; or
- iterate $z \leftarrow P^T z$ until convergence.

The latter is generally much less efficient, but may be the only option for very high dimensional Markov chains.

My approach will be to regard P and therefore everything else as depending on a packet loss probability p, so we do all calculations in computer algebra as formal (in fact, convergent) power series in p (see Appendix B). A basic principle of the whole approach will be that every quantity of practical interest (for example, goodput) is computable from the invariant distribution. However, we will also be interested in how the invariant distribution (and thus quantities dependent upon it), vary with the loss probability p. Thus, we will compute the full functional dependence of all quantities on p, not just produce approximations of the expectation value of these quantities as in most previous work.

4 Mini-TCP

All the ideas will be illustrated by the following toy model. Although it is trivial (in fact, exactly solvable), none of the essentials are lost. Exactly the same procedure will work for a full, realistic TCP. The advantage of first doing a toy model is that exact solutions are available for checking.

We let the window size at time k be w_k for $k = 0, 1, 2, \ldots$. Then mini-TCP adjusts its window size according to:

$$w_{k+1} = \begin{cases} \lceil w_k/2 \rceil & \text{if a packet is lost} \\ \min(w_k + 1, w_{max}) & \text{else} \end{cases}$$

with $w_{max} = 4$. The packet loss probability is p and we write $q = 1 - p$.

Thus, we have

$$P = \begin{bmatrix} p & q & & & \\ p & & q & & \\ & p & & q & \\ & p & & & q \\ & & p & & q \end{bmatrix}.$$

By computing the kernel of Q we obtain the exact formula for the expectation value of window size

$$E[w] = \frac{(1-p)(-4+7p-6p^2+2p^3)}{-1+2p-3p^2+p^3} = 4 - 3p - 5p^2 - 5p^3 + 4p^4 + \mathcal{O}(p^5).$$

My software deals with matrices with formal power series elements. This means that a truncation order must be set in advance, and all results are truncated to this order immediately after elementary operations such as addition, multiplication, etc. This is achieved in C++ by operator overloading. The full code is given in Appendix C. In outline, the procedure is:

1. Specify P and compute Q.

2. Compute z by iterating $z \leftarrow P^T z$. Since the formal power series have integer coefficients, when they cease to change, they will be correct. Thus a finite number of iterations will always suffice.

3. Normalize z.

4. Compute the fundamental matrix $Q^{\#}$ and thus sensitivity $\dot{z}$. $Q^{\#}$ may be computed from $Q^{\#} = \sum_{k=0}^{\infty}(P^k - [z\, z\, \ldots\, z]^T)$, where again the sum will be finite when all matrix elements are formal power series.

For the mini-TCP example, this procedure gives the invariant distribution as

$$\begin{aligned}
z(0) &= 2p^3 + 3p^4 - 7p^6 - 11p^7 + \dots \\
z(1) &= 2p^2 + p^3 - 3p^4 - 7p^5 - 4p^6 + 10p^7 + \dots \\
z(2) &= p - 2p^3 - 3p^4 + 7p^6 + 11p^7 + \dots \\
z(3) &= p - p^2 - 2p^3 - p^4 + 3p^5 + 7p^6 + 4p^7 + \dots \\
z(4) &= 1 - 2p - p^2 + p^3 + 4p^4 + 4p^5 - 3p^6 - 14p^7 + \dots .
\end{aligned}$$

and the expectation value of window size as

$$E[w] = 4 - 3p - 5p^2 - 5p^3 + 4p^4 + 18p^5 + 19p^6 - 12p^7 + \dots,$$

agreeing with the exact formula above. I also did a simulation as a second verification and obtained the graph of figure 2.

The matrices involved in this method can become very large but are sparse. The normally cannot be stored in full; however, in my C++ implementation it is quite simple to treat the matrices as functions of their two indices and just compute elements as needed; thus P_{ij} becomes a function $P(i,j)$.

5 Markov chain perturbation theory

Suppose the packet loss probability p changes with time; how does the invariant distribution change? The relevant theory to answer this question makes use of the related concepts of *fundamental matrix* and *group inverse* [Campbell & Meyer 1991, Meyer 1998] (see Appendix A of the present paper).

Suppose first that the matrix P is perturbed: $P \to P + \Delta = \tilde{P}$ (the perturbation is not necessarily small). How does the invariant distribution z change? Suppose z becomes $\tilde{z}$ after the perturbation. We have $z^T - \tilde{z}^T = \tilde{z}^T \Delta F$, and the following bounds are known [Meyer 1998]:

$$\begin{aligned}
||z^T - \tilde{z}^T|| &\leqslant ||\Delta||\,||F|| \\
\left|\frac{Q_{ij} - \widetilde{Q}_{ij}}{Q_{ij}}\right| < \epsilon \;\Rightarrow\; \left|\frac{z_j - \tilde{z}_j}{z_j}\right| &< 2n\epsilon + \mathcal{O}(\epsilon^2) \\
||z^T - \tilde{z}^T||_\infty &\leqslant ||\Delta||_\infty \; \mathrm{tr}(Q^\#) \\
||z_j^T - \tilde{z}_j^T|| &\leqslant ||\Delta||_\infty \; \max_i |Q_{ij}^\#|
\end{aligned}$$

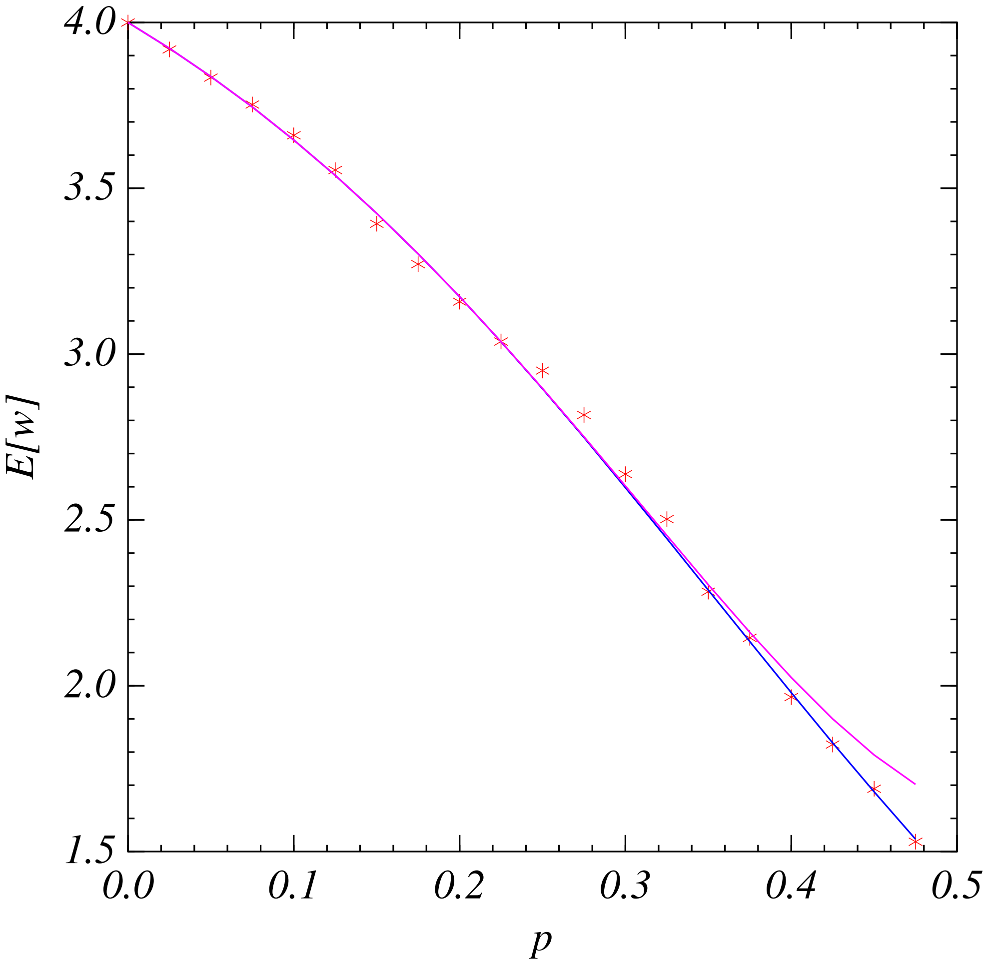

Figure 2: Expected window size: exact, series, simulation *

Here n is the dimension of the matrix P, and the unspecified norms are either the 1-norm or ∞-norm.

While these formulas are useful, we would ideally like precise formulas for the derivative $\dot{z}$ of z with respect to p. In our TCP application, this would provide information about the rate of adaption of the TCP algorithm to a changing environment. This derivative can be obtained from the formula [Meyer 1998]:

$$\dot{z}^T = z^T \dot{P} Q^{\#}$$

where the fundamental matrix: $F = (Q - \mathbf{1} z^T)^{-1}$ is the *group inverse* $Q^{\#}$ of Q.

Thus we can compute all required quantities. The C++ code is given in Appendix C. For example, the first row of $Q^{\#}$ with the mini-TCP example is:

$$\begin{array}{l} 1 + 2p + 5p^2 - 6p^3 + \mathcal{O}(p^4) \\ 1 + 3p - 9p^2 - 21p^3 + \mathcal{O}(p^4) \\ 1 - 4p - 7p^2 + 3p^3 + \mathcal{O}(p^4) \\ 1 - 6p - 2p^2 + 12p^3 + \mathcal{O}(p^4) \\ -5 + 5p + 13p^2 + 12p^3 + \mathcal{O}(p^4) \end{array}$$

Note, however, that F is dense. We can avoid density in practice by only computing products xF.

As an example of computation of $\dot{z}$ for mini-TCP: the derivative of z with respect to p is:

$$\begin{array}{rcl} \dot{z}(0) & = & 6p^2 + 12p^3 + \mathcal{O}(p^5) \\ \dot{z}(1) & = & 4p + 3p^2 - 12p^3 - 35p^4 + \mathcal{O}(p^5) \\ \dot{z}(2) & = & 1 - 6p^2 - 12p^3 + \mathcal{O}(p^5) \\ \dot{z}(3) & = & 1 - 2p - 6p^2 - 4p^3 + 15p^4 + \mathcal{O}(p^5) \\ \dot{z}(4) & = & -2 - 2p + 3p^2 + 16p^3 + 20p^4 + \mathcal{O}(p^5) \end{array}$$

6 NewReno TCP

Having developed the methods and verified them with mini-TCP, we now apply them to a realistic TCP algorithm, NewReno. This has the following state variables:

- $w_i, \quad 1 \leqslant w_i \leqslant w_{max}$: window size

- c_i, $\quad c_i = 0, 1$: $0 \Rightarrow$ first of two rounds; $1 \Rightarrow$ second

- l_i, $\quad 0 \leqslant w_i < w_{i-1}$: lost packets in previous round

- t_i, $\quad 0 \leqslant w_i \leqslant 7$: backoff factor in previous round was 2^{t_i-1}

The elements of the transition matrix P are given by three subcases. Firstly, if no packets are lost in a round:

$$
\begin{aligned}
P(w,0,0,0;w,1,0,0) &= q^w, \qquad 1 \leqslant w < w_{max} \\
P(w,1,0,0;w+1,0,0,0) &= q^w, \qquad 1 \leqslant w < w_{max} \\
P(w,0,0,0;w,0,0,0) &= q^w, \qquad w = w_{max}
\end{aligned}
$$

Secondly, if one or more packets are lost:

$$
\begin{aligned}
P(w,c,0,0;w-l,0,l,0) &= pq^{w-l}, \qquad 2 \leqslant w \leqslant w_{max}, 1 \leqslant l < w \\
P(w,c,0,0;1,0,0,1) &= p, \qquad 1 \leqslant w \leqslant w_{max} \\
P(w,0,l,0;1,0,0,1) &= 1, \qquad 1 \leqslant w < 3 \\
P(w,0,l,0;1,0,0,1) &= p\sum_{i=0}^{2} q^i, \qquad 3 \leqslant w < w_{max} \\
P(w,0,l,0;\lfloor (w+l)/2 \rfloor,0,0,1) &= q^w + p\sum_{i=3}^{w-1} q^i, \qquad 3 \leqslant w < w_{max}
\end{aligned}
$$

The exponential backoff rule (for $1 \leqslant i \leqslant 7$) is:

$$
\begin{aligned}
P(1,0,0,i;1,0,0,\min(i+1,7)) &= p \\
P(1,0,0,i;2,0,0,0) &= q
\end{aligned}
$$

With $w_{max} = 10$, this gives 71 allowed states out of a possible $10 \times 2 \times 9 \times 8 = 1440$ states, giving a sparsity of about 5%

Having specified the states and transition probabilities, the analysis by my method is fully automatic. However, the output just to describe the invariant distribution is very large, so I just give a few lines of typical output:

$$\begin{aligned}
z(8,0,0,0) &= 2.19p - 220.p^2 + 11832p^3 - 391235p^4 + O(p^5) \\
z(8,0,1,0) &= 4.38p^2 - 565p^3 + 37989p^4 + O(p^5) \\
z(8,0,2,0) &= 0.219p - 30.2p^2 + 2158p^3 - 100230p^4 + O(p^5) \\
z(8,1,0,0) &= 2.19p - 237p^2 + 13657p^3 - 492191p^4 + O(p^5) \\
z(9,0,0,0) &= 2.19p - 255p^2 + 15621p^3 - 608232p^4 + O(p^5) \\
z(9,0,1,0) &= 0.219p - 30.4p^2 + 2189p^3 - 102388p^4 + O(p^5) \\
z(9,1,0,0) &= 2.19p - 275p^2 + 17999p^3 - 758208p^4 + O(p^5) \\
z(10,0,0,0) &= 0.219 - 28.5p + 1924p^2 - 84022p^3 + O(p^4)
\end{aligned}$$

7 Mean first passage time

Let M be the matrix whose i,j element is the mean time for first transition from state i to state j. Then from [Campbell & Meyer 1991], page 159, we have

$$M = (I - Q^{\#} + \mathbf{1}\mathbf{1}^T \operatorname{diag}(Q^{\#}))(\operatorname{diag}(I - QQ^{\#}))^{-1}$$

This can be used, for example, to compute the expected start-up time, that is, the expected time to reach maximum window size from an initial size of one. For mini-TCP, this matrix to lowest order in p is

$$M = \begin{bmatrix} 5/2p^{-3} & 5/2p^{-2} & 5p^{-1} & 5p^{-1} & 9+18p \\ 3p^{-3} & 5/2p^{-2} & 5p^{-1} & 5p^{-1} & 8+17p \\ 3p^{-3} & 3p^{-2} & 5p^{-1} & 5p^{-1} & 7+15p \\ 3p^{-3} & 3p^{-2} & 6p^{-1} & 5p^{-1} & 6+13p \\ 3p^{-3} & 3p^{-2} & 6p^{-1} & 6p^{-1} & 5+10p \end{bmatrix}.$$

8 Conclusion

With this method, we can now analyze more-or-less automatically any TCP version. However, there is still much work to be done. It would be nice to have a simple interface (a kind of TCP specification language) which could automatically produce a full analysis of a proposed TCP variant. This could then be used to optimize TCP performance for a given application.

A Generalized inverses of matrices

Suppose A is a real matrix that is square and singular, or non-square. Then it has no ordinary inverse, but various pseudo-inverses may used to provide a solution to a least-squares problem, or to provide an inverse on some subspace. The most important are these three, which are uniquely defined by the given conditions.

The *Moore-Penrose pseudo-inverse* $A^\dagger$ is defined for all matrices A:

$$\begin{aligned} AA^\dagger A &= A \\ A^\dagger AA^\dagger &= A^\dagger \\ (AA^\dagger)^T &= AA^\dagger \\ (A^\dagger A)^T &= A^\dagger A \end{aligned}$$

The *Drazin pseudo-inverse* A^D is defined for all $n \times n$ square matrices A for which there exists a minimal non-negative integer k (the *index* of A) such that $\operatorname{rank} A^k = \operatorname{rank} A^{k+1}$:

$$\begin{aligned} A^D AA^D &= A^D \\ AA^D &= A^D A \\ A^k A^D &= A^k \end{aligned}$$

The Drazin pseudo-inverse in the special case where either $\operatorname{rank}(A) = n$ or $\operatorname{rank}(A) = \operatorname{rank}(A^2)$ becomes the *group inverse* $A^\#$ which satisfies:

$$\begin{aligned} AA^\# A &= A \\ A^\# AA^\# &= A^\# \\ AA^\# &= A^\# A \end{aligned}$$

It is this last pseudo-inverse that is most useful for studying Markov chains.

B Formal power series

All of my code makes use of representations of quantities as formal power series in p. These formal power series are just sequences with elements in the field $\mathbb{R}$. The set of such formal power series is denoted $\mathbb{R}[[x]]$. I write either $a = [a_0, a_1, \ldots]$ or $\sum_{i=0}^{\infty} a_i x^i$, where $a_i \in \mathbb{R}$. The second notation is a formal device, and does not imply that x has any value, or that there is any infinite sum. The first element a_0 of the sequence is called the constant term. $\mathbb{R}[[x]]$ is an integral domain (a ring without zero divisors). The operations are termwise addition and Cauchy multiplication ($c = ab \iff c_j = \sum_{i=0}^{j} a_i b_{j-i}$). Units (that is, invertible elements) are formal power series with nonzero constant term. These are formal power series having multiplicative inverses. *Series composition*, denoted $a \circ b$, is the substitution of b for x in a. The compositional identity formal power series is $[0, 1] \equiv x$. For $a \circ b$ to be well-defined, we require $b_0 = 0$, that is, b must be a nonunit. Any nonunit b has a compositional inverse (or reversion), denoted $b^{<-1>}$. It of course satisfies $b^{<-1>} \circ b = b \circ b^{<-1>} = [0, 1]$.

C The code for mini-TCP

```
// Keith Briggs 2002 June 10
// GiNaC fps for mini-TCP

#include <iostream>
#include <stdexcept>
#include <ginac/ginac.h>

using namespace std;
using namespace GiNaC;
typedef matrix colvec;

matrix fundamental(const matrix& P, const colvec& z);

const int dbg=0,n=5,O=5;
symbol p("p");

int main() {
  int i,j;
  matrix A(n,n),P(n,n),F(n,n),Q(n,n),Pdot(n,n),Qsharp(n,n);
  colvec z(n,1),zz(n,1),zdotT(n,1),c(n,1);
  ex q,e,norm,s;
  q=1-p;
  // P...
  P(0,0) = p; P(0,1) = q; P(0,2) = 0; P(0,3) = 0; P(0,4) = 0;
  P(1,0) = p; P(1,1) = 0; P(1,2) = q; P(1,3) = 0; P(1,4) = 0;
  P(2,0) = 0; P(2,1) = p; P(2,2) = 0; P(2,3) = q; P(2,4) = 0;
  P(3,0) = 0; P(3,1) = p; P(3,2) = 0; P(3,3) = 0; P(3,4) = q;
  P(4,0) = 0; P(4,1) = 0; P(4,2) = p; P(4,3) = 0; P(4,4) = q;
  // Q=I-P...
  for (i=0; i<n; i++) for (j=0; j<n; j++) if (i==j) Q(i,j)=1-P(i,j); else Q(i,j)=-P(i,j);
  // dP/dp...
  for (i=0; i<n; i++) for (j=0; j<n; j++) Pdot(i,j)=P(i,j).diff(p,1);
  for (i=0; i<n; i++) c(i,0)=1;
  cout<<"P="<<P<<endl;
  cout<<"Q="<<Q<<endl;
  cout<<"Pdot="<<Pdot<<endl;
  for (i=0; i<n; i++) z(i,0)=1;
  do { // find stationary distbn z...
    zz=z;
    z=P.transpose().mul(z);
    if (dbg) cout<<z<<endl;
    for (j=0; j<n; j++) z(j,0)=z(j,0).series(p,O);
  } while (z!=zz);
  for (s=i=0; i<n; i++) s+=z(i,0);
  for (i=0; i<n; i++) z(i,0)=(z(i,0)/s).series(p==0,O);
  cout<<"\nInvariant distribution="<<endl;
  for (i=0; i<n; i++) cout<<"z("<<i<<")="<<z(i,0)<<endl;
  for (s=i=0; i<n; i++) s+=i*z(i,0);
  cout<<"E[w]="<<s.series(p==0,O)<<endl;
  F=fundamental(P,z);
  Qsharp=F;
  cout<<"Qsharp="<<Qsharp<<endl;
  A=(F.mul(Q)).add(c.mul(z.transpose())); // F*Q=I-c*z^T ?
  cout<<"Should be I..."<<endl;
  for (i=0; i<n; i++) for (j=0; j<n; j++) cout<<A(i,j).series(p==0,O)<<endl;
  if (dbg) cout<<"Pdot.mul(Qsharp)="<<Pdot.mul(Qsharp)<<endl;
  zdotT=(z.transpose().mul(Pdot.mul(Qsharp))).transpose(); // z^T.=z^T*P.Q#
  for (i=0; i<n; i++) zdotT(i,0)=zdotT(i,0).series(p==0,O);
  cout<<"z dot ..."<<endl;
  for (i=0; i<n; i++) cout<<i<<" "<<zdotT(i,0)<<endl;
  zdotT.print(print_latex(std::cout));
  return 0;
}

matrix fundamental(const matrix& P, const colvec& z) {
  int i,j;
  ex f00;
  matrix F(n,n),FF(n,n),Pk(n,n),A(n,n);
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) A(i,j)=z(j,0), Pk(i,j)=F(i,j)=0;
    Pk(i,i)=1;
  }
  do {
    FF=F;
    F=F.add(Pk.sub(A));
    Pk=Pk.mul(P);
    for (i=0; i<n; i++) for (j=0; j<n; j++) F(i,j)=F(i,j).series(p==0,O);
  } while (FF!=F);
  return F;
}
```

References

Campbell, S. L. & Meyer, C. D. [1991], Generalized inverses of linear transformations, Dover.

Dumas, V., Guillemin, F. & Robert, P. [no date], Limit results for Markovian models of TCP, Technical report, Submitted to Internet Performance Symposium.

Jacobson, V. & Karels, M. K. [1988], Congestion avoidance and control, Technical report, Lawrence Berkeley Laboratory.

Meyer, C. D. [1998], Sensitivity of Markov chains, Technical report, North Carolina State University, 7th Conference of the International Linear Algebra Society.

Padhye, J., Firoiu, V. & Towsley, D. [1999], A stochastic model of TCP Reno congestion avoidance and control, Technical report, Department of Computer Science, University of Massachusetts. CMPSCI Technical report 99-02.

Padhye, J., Firoiu, V., Towsley, D. & Kurose, J. [1998], Modeling TCP throughput: a simple model and its empirical validation, Technical report, Department of Computer Science, University of Massachusetts. ACM SIGCOMM'98.