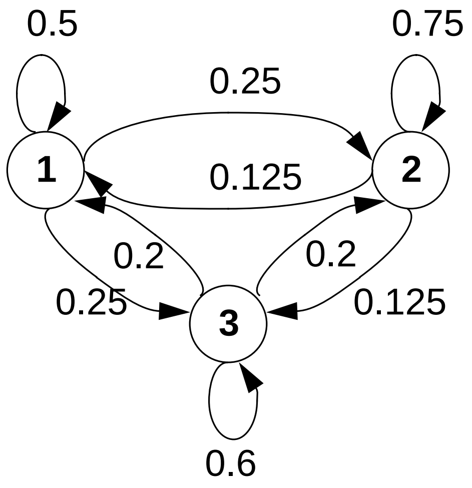
An Example to Illustrate The Concept of Entropy Rate

A three-state Markov process is shown below. Its transition probability matrix is given as:

$$P = \begin{bmatrix} 0.500 & 0.250 & 0.250 \\ 0.125 & 0.750 & 0.125 \\ 0.200 & 0.200 & 0.600 \end{bmatrix} \tag{1}$$

We would like to compute the stationary state probability distribution.



Let the initial state probability be represented as:

$$\underline{\mu} = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 \end{bmatrix} \tag{2}$$

If the distribution is stationary, then $\underline{\mu} = \underline{\mu} \cdot P$. We can solve these simultaneous equations to get the stationary distribution.

If the initial distribution is not the stationary distribution, the state distribution at a time n, is given in terms of P the state transition matrix and the initial state distribution, μ_o as:

$$\underline{\mu_n} = \underline{\mu_0} \cdot P^n \tag{3}$$

Therefore, using the state distribution at time n, we can find the entropy at time instant n of the Markov random process X as:

$$H(X) = \sum_i \mu_i \cdot \log\left(\frac{1}{\mu_i}\right) \tag{4}$$

Note that, by definition, if we initialize the process with the stationary distribution, the entropy at all times is constant and equal to the entropy rate. If we do not start with the stationary distribution the entropy will converge to the entropy rate with time (with possible oscillatory behavior).

Solving for the stationary distribution we get:

$$\underline{\mu_0} = \begin{bmatrix} 0.235 & 0.470 & 0.295 \end{bmatrix} \tag{5}$$

To verify that this indeed is the stationary distribution, let us compute the state distribution at time $n = 1$.

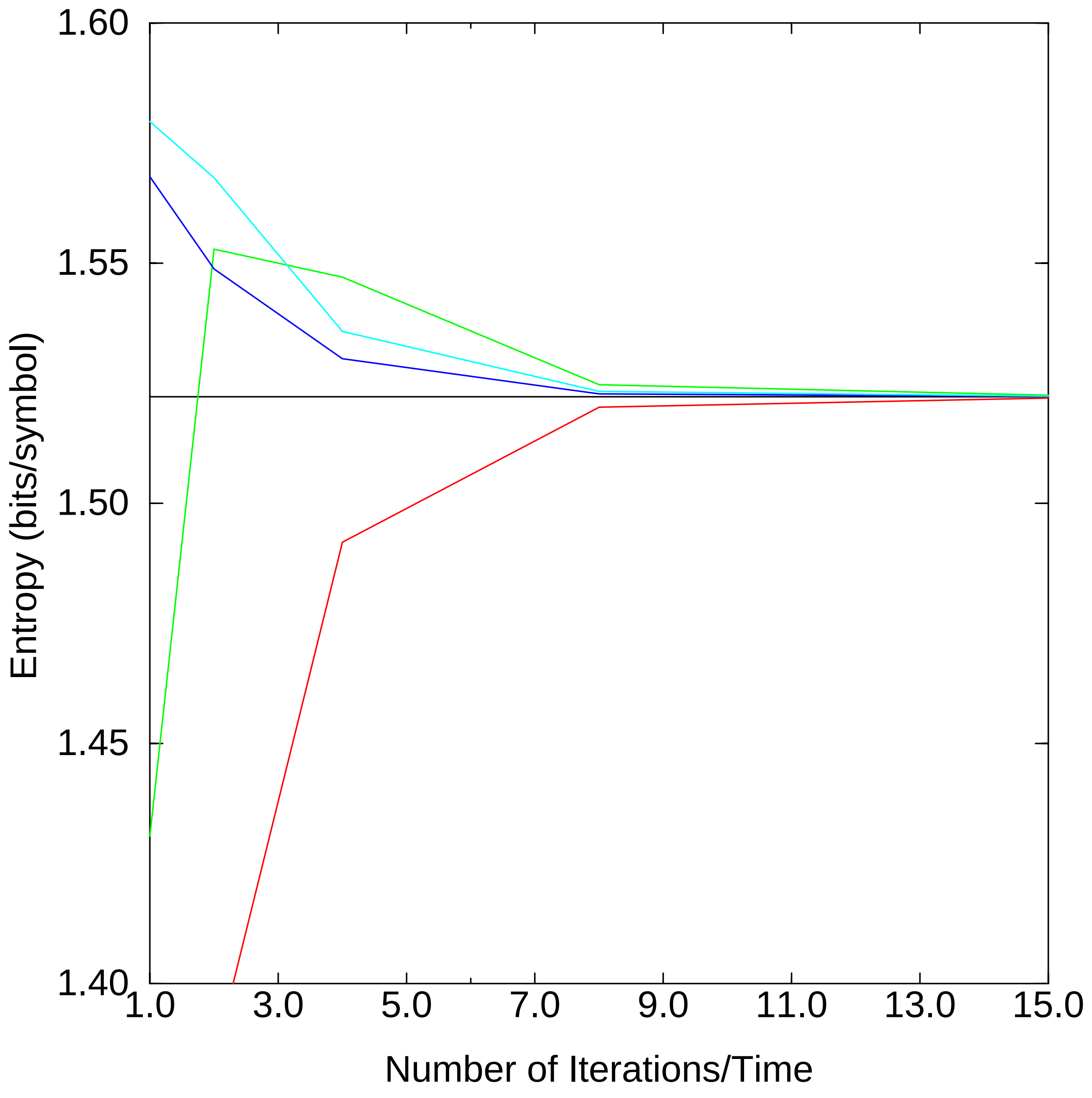
$$\begin{aligned} \underline{\mu_1} &= \underline{\mu} \cdot P^1 \\ &= \begin{bmatrix} 0.235 & 0.470 & 0.295 \end{bmatrix} \cdot \begin{bmatrix} 0.500 & 0.250 & 0.250 \\ 0.125 & 0.750 & 0.125 \\ 0.200 & 0.200 & 0.600 \end{bmatrix} \\ &= \begin{bmatrix} 0.235 & 0.470 & 0.295 \end{bmatrix} \end{aligned} \tag{6}$$

The entropy at time $n = 1$ is found using Eq. 4:

$$H(X)\big|_{n=1} = 1.522190 \tag{7}$$

Since we started with the stationary distribution, we know that this is the same as the entropy rate.

The following plot illustrates the behavior of the entropy as a function of time when the initial probability distribution is not the stationary distribution. Each curve on the plot has an initial distribution which is different from the stationary distribution. Note that whatever the distribution we start with the entropy finally converges to the entropy rate. Note the oscillatory behaviour of some of the curves.



The code used to generate the at for this plot follows follows this plot.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// this program generates data to illustrate the effect of the initial
// conditions on the entropy rate of a markov process
//
main (int argc, char** argv) {

  // check for usage
  //
  if (argc < 2) {
    fprintf (stdout, “Usage: entropy_rate <num_states> <num_iterations>\n”);
  }

  // define local variables
  //
  int num_rows = atoi(argv[1]);
  int num_cols = num_rows;
  float entropy, sum, entropy_rate;
  float** a = new float*[num_rows];
  for (int kk = 0; kk < num_rows; kk++) {
    a[kk] = new float[num_cols];
  }

  float** a_new = new float*[num_rows];
  for (int kk = 0; kk < num_rows; kk++) {
    a_new[kk] = new float[num_cols];
  }

  float* u = new float[num_cols];
  float* u_new = new float[num_cols];

  // open files for i/o
  //
  FILE* fp_trans = fopen (argv[3],”r”);
```

```
// read the transition probalilities from the file
//
for (int kk = 0; kk < num_rows; kk++) {
  for (int jj = 0; jj < num_cols; jj++) {
    fscanf (fp_trans, “%f”, &a[kk][jj]);
    a_new[kk][jj] = a[kk][jj];
  }
}

// read the initial probabilities
//
for (int kk = 0; kk < num_cols; kk++) {
  fscanf (fp_trans, “%f”, &u[kk]);
}

// compute the new state probs from the initial probs
//

// first compute a.exp(t-1)
//
int num_loops = (int)(log(atoi(argv[2]))/log(2));
for (int tt = 0; tt < num_loops; tt++) {
  for (int jj = 0; jj < num_cols; jj++) {
    for (int kk = 0; kk < num_rows; kk++) {
sum = 0;
for (int j = 0 ; j < num_cols; j++) {
  sum +=  a[j][kk]*a[jj][j];
}
a_new[jj][kk] = sum;
    }
  }

  for (int k = 0; k < num_rows; k++) {
    for (int j = 0; j < num_cols; j++) {
a[k][j] = a_new[k][j];
    }
  }
}
```

```
  // compute the new state probabilities
  //
  for (int k = 0; k < num_cols; k++) {
    for (int j = 0; j < num_rows; j++) {
      u_new[k] += u[j]*a_new[j][k];
    }
  }

  // compute the entropy at this instant in time
  //
  entropy = 0;
  for (int k = 0; k < num_cols; k++) {
    entropy += u_new[k] * (log10(1/u_new[k])/ log10(2.0));
  }
  fprintf (stdout,"%d %1.4f \n", atoi(argv[2]),entropy);

  // compute the entropy rate
  //
  entropy_rate = 0.0;
  for (int k = 0; k < num_rows; k++) {
    for (int j = 0; j < num_cols; j++) {
      entropy_rate -= u_new[k]*a[k][j]*(log10(a[k][j])/log10(2)) ;
    }
  }

  // exit gracefully
  //
}
```