Hastings - Metropolis Algorithm implementation

by Naser Abbasi. Mathematics 504, Spring 2008. CSUF

This below is an implementation of the Hastings - Metropolis algorithm. A simple GUI interface allows the user to specify the number of steps to run the algorithm for. At each step, the current P matrix and the current calculated stationary distribution for this P matrix are shown to help observe the convergence.

The input to this run below is that of example 8.3.1 from Lecture notes of Math 504 by Professor B. Gearhart, CSUF. Below I show the user interface, and few screen shots showing the progress of the convergence to the final P matrix.

Few seed the random number generator and display the q and the π distribution used

In[1]:=
```
SeedRandom[121212];
MatrixForm[q = {{0, .5, .5}, {.5, 0, .5}, {.5, .5, 0}}]
```

Out[2]//MatrixForm=

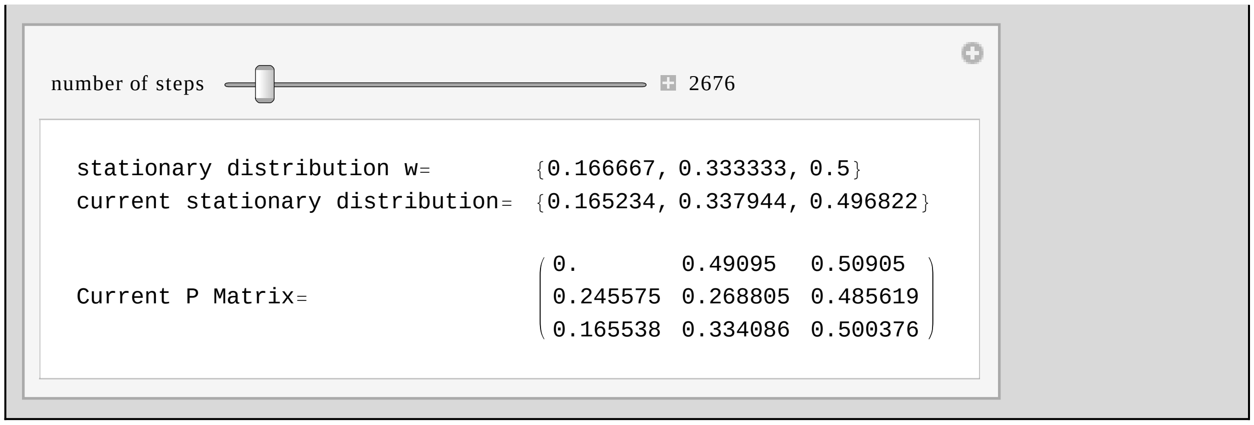
$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix}$$

In[3]:=
```
MatrixForm[pi = {1/6, 1/3, 1/2}]
```

Out[3]//MatrixForm=

$$\begin{pmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{2} \end{pmatrix}$$

```
In[9]:= m = Manipulate[
          First@{x = hastings[pi, q, maxN]; Grid[{{"stationary distribution w=", N[pi]},
              { "current stationary distribution=", MatrixPower[N[x], 100][[1, All]]},
              {, }, {"Current P Matrix=", N[MatrixForm[x]]}}, Alignment → Left]},
          {{maxN, 1, "number of steps"}, 1, 50 000, 25, ContinuousAction → False,
           Appearance → "Labeled"},
          AutorunSequencing → {{1, 300}}
         ]
```

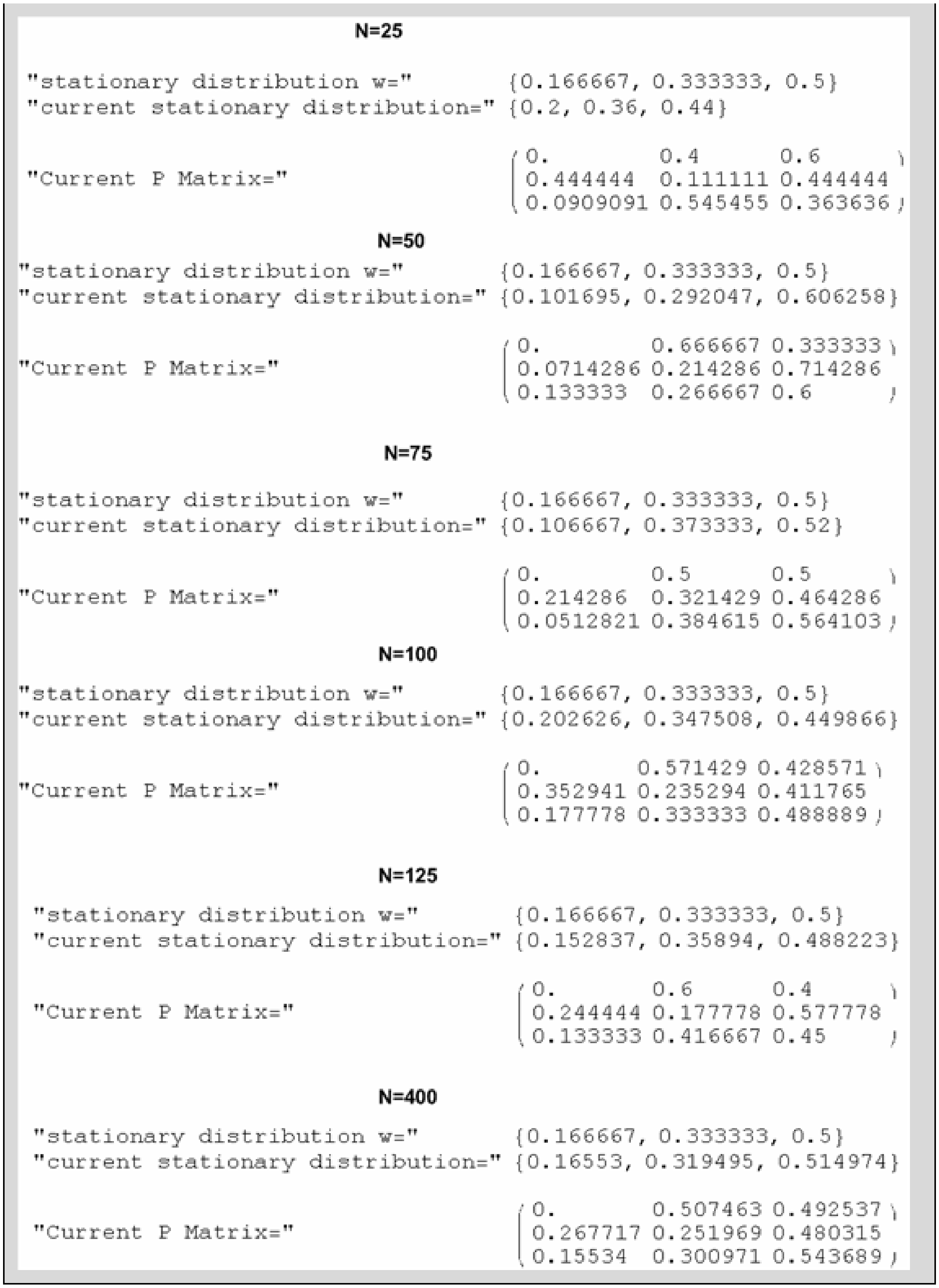


```
Out[9]=
number of steps    2676

stationary distribution w=         {0.166667, 0.333333, 0.5}
current stationary distribution=   {0.165234, 0.337944, 0.496822}

                                    ⎛0.        0.49095   0.50905 ⎞
Current P Matrix=                   ⎜0.245575  0.268805  0.485619⎟
                                    ⎝0.165538  0.334086  0.500376⎠
```

This a snap shot of the above output for number of iterations



N=25

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.2, 0.36, 0.44}

                                     ⎛0.         0.4      0.6     ⎞
"Current P Matrix="                  ⎜0.444444  0.111111 0.444444 ⎟
                                     ⎝0.0909091 0.545455 0.363636 ⎠
```

N=50

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.101695, 0.292047, 0.606258}

                                     ⎛0.         0.666667 0.333333⎞
"Current P Matrix="                  ⎜0.0714286 0.214286 0.714286 ⎟
                                     ⎝0.133333  0.266667 0.6      ⎠
```

N=75

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.106667, 0.373333, 0.52}

                                     ⎛0.         0.5      0.5     ⎞
"Current P Matrix="                  ⎜0.214286  0.321429 0.464286 ⎟
                                     ⎝0.0512821 0.384615 0.564103 ⎠
```

N=100

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.202626, 0.347508, 0.449866}

                                     ⎛0.        0.571429 0.428571⎞
"Current P Matrix="                  ⎜0.352941 0.235294 0.411765 ⎟
                                     ⎝0.177778 0.333333 0.488889 ⎠
```

N=125

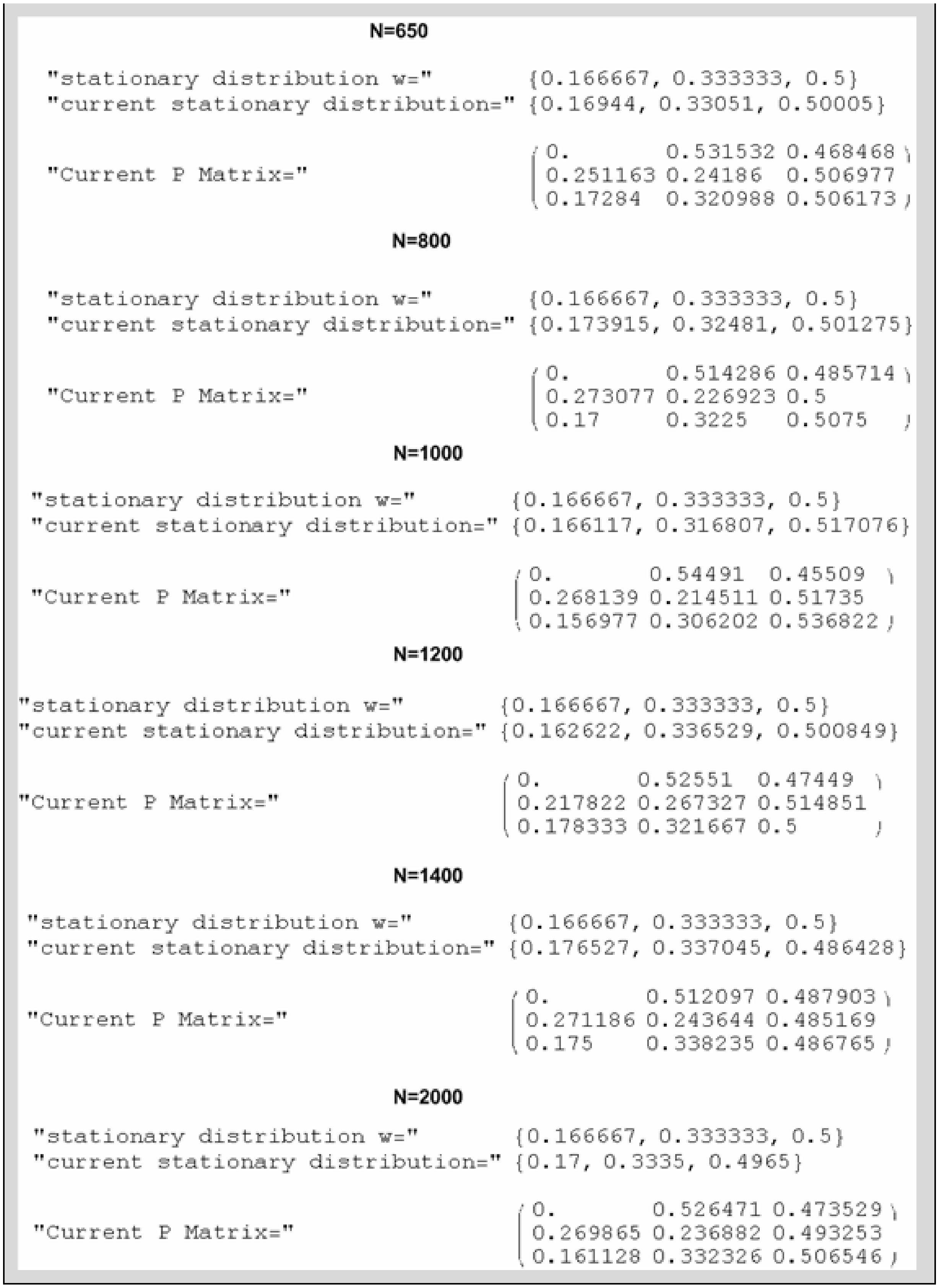
```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.152837, 0.35894, 0.488223}

                                     ⎛0.        0.6      0.4     ⎞
"Current P Matrix="                  ⎜0.244444 0.177778 0.577778 ⎟
                                     ⎝0.133333 0.416667 0.45     ⎠
```

N=400

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution="   {0.16553, 0.319495, 0.514974}

                                     ⎛0.        0.507463 0.492537⎞
"Current P Matrix="                  ⎜0.267717 0.251969 0.480315 ⎟
                                     ⎝0.15534  0.300971 0.543689 ⎠
```



N=650

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.16944, 0.33051, 0.50005}

                                   ⎛0.        0.531532 0.468468⎞
"Current P Matrix="                ⎜0.251163 0.24186  0.506977⎟
                                   ⎝0.17284  0.320988 0.506173⎠
```

N=800

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.173915, 0.32481, 0.501275}

                                   ⎛0.        0.514286 0.485714⎞
"Current P Matrix="                ⎜0.273077 0.226923 0.5     ⎟
                                   ⎝0.17     0.3225   0.5075  ⎠
```

N=1000

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.166117, 0.316807, 0.517076}

                                   ⎛0.        0.54491  0.45509 ⎞
"Current P Matrix="                ⎜0.268139 0.214511 0.51735 ⎟
                                   ⎝0.156977 0.306202 0.536822⎠
```

N=1200

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.162622, 0.336529, 0.500849}

                                   ⎛0.        0.52551  0.47449 ⎞
"Current P Matrix="                ⎜0.217822 0.267327 0.514851⎟
                                   ⎝0.178333 0.321667 0.5     ⎠
```

N=1400

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.176527, 0.337045, 0.486428}

                                   ⎛0.        0.512097 0.487903⎞
"Current P Matrix="                ⎜0.271186 0.243644 0.485169⎟
                                   ⎝0.175    0.338235 0.486765⎠
```

N=2000

```
"stationary distribution w="         {0.166667, 0.333333, 0.5}
"current stationary distribution=" {0.17, 0.3335, 0.4965}

                                   ⎛0.        0.526471 0.473529⎞
"Current P Matrix="                ⎜0.269865 0.236882 0.493253⎟
                                   ⎝0.161128 0.332326 0.506546⎠
```

Appending

Algorithm implementation

In[4]:=
```
cumSum[list_] := Module[{i, sum, s, k},
  sum = 0;
  k = Length[list];
  s = Table[0, {k}];
  For[i = 1, i ≤ k, i++,
   {
    sum = sum + list⟦i⟧;
    s⟦i⟧ = sum;
   }
  ];
  s
 ]
```

Function to calculate β (x, y)

In[5]:=
```
beta[x_, y_, pi_, q_] := Module[{},
  Min[1, (pi⟦y⟧ q⟦y, x⟧) / (pi⟦x⟧ q⟦x, y⟧)]
 ]
```

Function called at the end of the run to generate P from the path of states travelled

In[6]:=
```
generatePMatrixFromStatePath[nStates_, x_] := Module[{i, j, p, allPairs, n, m},
  n = Length[x];
  p = Table[0, {nStates}, {nStates}];
  allPairs = Partition[x, 2, 1];
  For[i = 1, i ≤ nStates, i++,
   {
    m = Count[allPairs, {i, y_}];
    For[j = 1, j ≤ nStates, j++,
     If[m ≠ 0, p⟦i, j⟧ = Count[allPairs, {i, j}] / m, p⟦i, j⟧ = 0]
    ]
   }
  ];

  p
 ]
```

Function to sample from q using uniform distribution

In[7]:=
```
sampleFromQConditional[q_, x_] := Module[{s, found, j, k, sample, y},
  s = Flatten[Position[q[[x, All]], Except[0], 1, Heads → False] ];
  sample = q[[ x, s]];
  sample = cumSum[sample];
  y = RandomReal[];
  found = False;
  For[j = 1, j ≤ Length[sample], j ++,
   If[ Not[found], If[y ≤ sample[[j]], {k = j; found = True}]]
  ];

  y = s[[k]]
 ]
```

The Hastings algorithm main loop

In[8]:=
```
hastings[pi_, q_, maxN_] :=
 Module[{i, j, nStates, n, s, y, α, u, x, sample, pts, sum, k, found},
  nStates = Length[q];
  n = 1;
  x = Table[0, {maxN}];
  x[[n]] = 1;  (*pick any state to start from*)
  i = 1;
  While[i < maxN,
   {
    y = sampleFromQConditional[q, x[[n]]];
    α = beta[x[[n]], y, pi, q];
    u = RandomReal[];
    n ++;
    If[u ≤ α, x[[n]] = y, x[[n]] = x[[n - 1]]]; (*acceptance step*)
    i ++;
   }
  ];

  generatePMatrixFromStatePath[nStates, x]
 ]
```