Section 2.10

MATLAB

Example 2.42 Problem

In Example 2.29, the random variable X, the number of pages in a fax, has PMF

$$P_X(x) = \begin{cases} 0.15 & x = 1, 2, 3, 4, \\ 0.1 & x = 5, 6, 7, 8, \\ 0 & \text{otherwise.} \end{cases}$$

Write a MATLAB function that calculates $P_X(x)$. Calculate the probability of x_i pages for $x_1 = 2$, $x_2 = 2.5$, and $x_3 = 6$.

Example 2.42 Solution

The MATLAB function `fax3pmf(x)` implements $P_X(x)$. We can then use `fax3pmf` to calculate the desired probabilities:

```
function y=fax3pmf(x)
s=(1:8)';
p=[0.15*ones(4,1); 0.1*ones(4,1)];
y=finitepmf(s,p,x);
```

```
>> fax3pmf([2 2.5 6])'
ans =
   0.1500   0   0.1000
```

Example 2.43 Problem

Write a MATLAB function `geometricpmf(p,x)` to calculate $P_X(x)$ for a geometric (p) random variable.

Example 2.43 Solution

```
function pmf=geometricpmf(p,x)
%geometric(p) rv X
%out: pmf(i)=Prob[X=x(i)]
x=x(:);
pmf= p*((1-p).^(x-1));
pmf= (x>0).*(x==floor(x)).*pmf;
```

In `geometricpmf.m`, the last line ensures that values $x_i \notin S_X$ are assigned zero probability. Because `x=x(:)` reshapes `x` to be a column vector, the output `pmf` is always a column vector.

Example 2.44 Problem

Write a MATLAB function that calculates the PMF of a Poisson (α) random variable.

Example 2.44 Solution

For an integer x, we could calculate $P_X(x)$ by the direct calculation

```
px= ((alpha^x)*exp(-alpha*x))/factorial(x)
```

This will yield the right answer as long as the argument x for the factorial function is not too large. In MATLAB version 6, `factorial(171)` causes an overflow. In addition, for $a > 1$, calculating the ratio $a^x/x!$ for large x can cause numerical problems because both a^x and $x!$ will be very large numbers, possibly with a small quotient. Another shortcoming of the direct calculation is apparent if you want to calculate $P_X(x)$ for the set of possible values $\mathbf{x} = [0, 1, \ldots, n]$. Calculating factorials is a lot of work for a computer and the direct approach fails to exploit the fact that if we have already

Example 2.45 Problem

Write a MATLAB function that calculates the CDF of a Poisson random variable.

Example 2.45 Solution

```
function cdf=poissoncdf(alpha,x)
%output cdf(i)=Prob[X<=x(i)]
x=floor(x(:));
sx=0:max(x);
cdf=cumsum(poissonpmf(alpha,sx));
  %cdf from 0 to max(x)
okx=(x>=0);%x(i)<0 -> cdf=0
x=(okx.*x);%set negative x(i)=0
cdf= okx.*cdf(x+1);
  %cdf=0 for x(i)<0
```

Here we present the MATLAB code for the Poisson CDF. Since a Poisson random variable X is always integer valued, we observe that $F_X(x) = F_X(\lfloor x \rfloor)$ where $\lfloor x \rfloor$, equivalent to `floor(x)` in MATLAB, denotes the largest integer less than or equal to x.

Example 2.46 Problem

Recall in Example 2.19 that a website has on average $\lambda = 2$ hits per second. What is the probability of no more than 130 hits in one minute? What is the probability of more than 110 hits in one minute?

Example 2.46 Solution

Let M equal the number of hits in one minute (60 seconds). Note that M is a Poisson (α) random variable with $\alpha = 2 \times 60 = 120$ hits. The PMF of M is

$$P_M(m) = \begin{cases} (120)^m e^{-120}/m! & m = 0, 1, 2, \ldots \\ 0 & \text{otherwise.} \end{cases}$$

```
>> poissoncdf(120,130)
ans =
    0.8315
>> 1-poissoncdf(120,110)
ans =
    0.8061
```

The MATLAB solution shown on the left executes the following math calculations:

$$P[M \leq 130] = \sum_{m=0}^{130} P_M(m)$$

$$P[M > 110] = 1 - P[M \leq 110]$$

$$= 1 - \sum_{m=0}^{110} P_M(m)$$

Example 2.47 Problem

Write a MATLAB function that generates m samples of a binomial (n, p) random variable.

Example 2.47 Solution

```
function x=binomialrv(n,p,m)
% m binomial(n,p) samples
r=rand(m,1);
cdf=binomialcdf(n,p,0:n);
x=count(cdf,r);
```

For vectors `x` and `y`, the function `c=count(x,y)` returns a vector `c` such that `c(i)` is the number of elements of `x` that are less than or equal to $y(i)$.

In terms of our earlier pseudocode, $k^* =$ `count(cdf,r)`.
If `count(cdf,r)` $= 0$, then $r \leq P_X(0)$ and $k^* = 0$.

Example 2.48 Problem

Simulate $n = 1000$ trials of the experiment producing the power measurement Y in Example 2.30. Compare the relative frequency of each $y \in S_Y$ to $P_Y(y)$.

Example 2.48 Solution

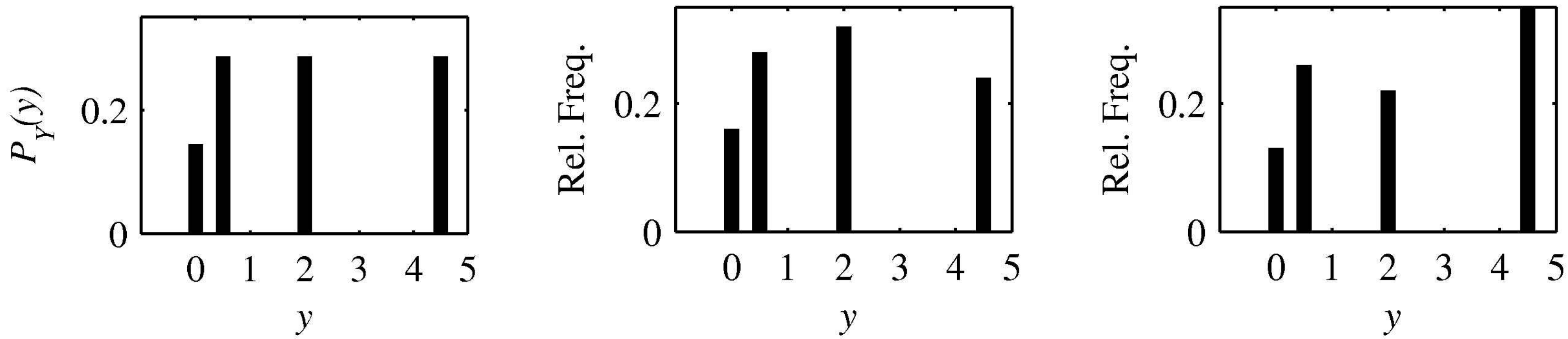
In `voltpower.m`, we first generate n samples of the voltage V. For each sample, we calculate $Y = V^2/2$.

```
function voltpower(n)
v=duniformrv(-3,3,n);
y=(v.^2)/2;
yrange=0:max(y);
yfreq=(hist(y,yrange)/n)';
pmfplot(yrange,yfreq);
```

As in Example 1.47, the function `hist(y,yrange)` produces a vector with jth element equal to the number of occurences of `yrange(j)` in the vector `y`. The function `pmfplot.m` is a utility for producing PMF bar plots in the style of this text.

Figure 2.2 shows the corresponding PMF along with the output of two runs of `voltpower(100)`.

Figure 2.2



PMF $P_Y(y)$ Sample Run 1 Sample Run 2

The PMF of Y and the relative frequencies found in two sample runs of `voltpower(100)`. Note that in each run, the relative frequencies are close to (but not exactly equal to) the corresponding PMF.

Example 2.49

```
>> sx=[1 3 5 7 3];
>> px=[0.1 0.2 0.2 0.3 0.2];
>> pmfx=finitepmf(sx,px,1:7);
>> pmfx'
ans =
   0.10 0 0.40 0 0.20 0 0.30
```

The function `finitepmf()` accounts for multiple occurrences of a sample value. In particular,

`pmfx(3)=px(2)+px(5)=0.4`.

Example 2.50 Problem

Recall that in Example 2.29 that the number of pages X in a fax and the cost $Y = g(X)$ of sending a fax were described by

$$P_X(x) = \begin{cases} 0.15 & x = 1, 2, 3, 4, \\ 0.1 & x = 5, 6, 7, 8, \\ 0 & \text{otherwise,} \end{cases} \qquad Y = \begin{cases} 10.5X - 0.5X^2 & 1 \le X \le 5, \\ 50 & 6 \le X \le 10. \end{cases}$$

Use MATLAB to calculate the PMF of Y.

Example 2.50 Solution

```
%fax3y.m
sx=(1:8)';
px=[0.15*ones(4,1); ...
        0.1*ones(4,1)];
gx=(sx<=5).* ...
     (10.5*sx-0.5*(sx.^2))...
   + ((sx>5).*50);
sy=unique(gx);
py=finitepmf(gx,px,sy);
```

The vector `gx` is the mapping $g(x)$ for each $x \in S_X$. In `gx`, the element 50 appears three times, corresponding to $x = 6$, $x = 7$, and $x = 8$. The function `sy=unique(gx)` extracts the unique elements of `gx` while `finitepmf(gx,px,sy)` calculates the probability of each element of `sy`.