6.7 EXAMPLES OF MATLAB APPLICATIONS

Sample Problem 6-9: Withdrawing from a retirement account.

A person in retirement is depositing $300,000 in a saving account that pays 5% interest per year. The person plans to withdraw money from the account once a year. He starts by withdrawing $25,000 after the first year, and in future years he increases the amount he withdraws according to the inflation rate. For example, if the inflation rate is 3%, he withdraws $25,750 after the second year. Calculate the number of years the money in the account will last assuming a constant yearly inflation rate of 2%. Make a plot that shows the yearly withdrawals and the balance of the account over the years.

Solution

The problem is solved by using a loop (a `while` loop since the number of passes is not known before the loop starts). In each pass the amount to be withdrawn and the account balance are calculated. The looping continues as long as the account balance is larger than or equal to the amount to be withdrawn. The following is a program in a script file that solves the problem. In the program, `year` is a vector in which each element is a year number, `W` is a vector with the amount withdrawn each year, and `AB` is a vector with the account balance each year.

```
rate=0.05; inf=0.02;
clear W AB year
year(1)=0;
W(1)=0;
AB(1)=300000;
Wnext=25000;
ABnext=300000*(1 + rate);
n=2;
    while ABnext >= Wnext
        year(n)=n-1;
        W(n)=Wnext;
        AB(n)=ABnext-W(n);
        ABnext=AB(n)*(1+rate);
        Wnext=W(n)*(1+inf);
        n=n+1;
    end
fprintf('The money will last for %f years',year(n-1))
bar(year,[AB' W'],2.0)
```

First element is year 0.

Initial withdrawal amount.

Initial account balance.

The amount to be withdrawn after a year.

The account balance after a year.

`while` checks if the next balance is larger than the next withdrawal.

Amount withdrawn in year $n-1$.

Account balance in year $n-1$ after withdrawal.

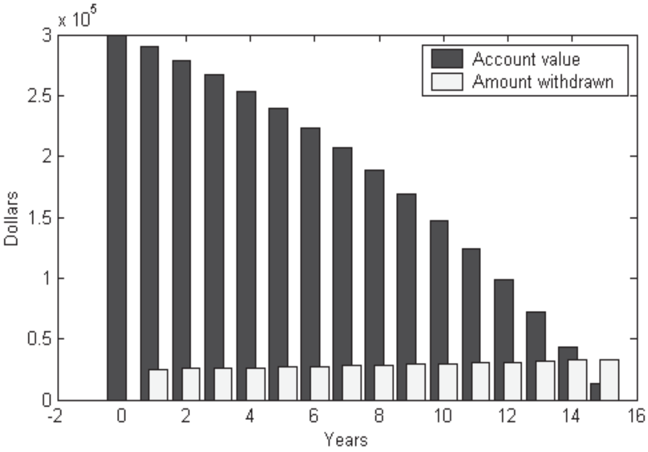
The account balance after additional year.

The amount to be withdrawn after an additional year.

The program is executed in the following Command Window:

```
>> Chap6_exp9
The money will last for 15 years.
```

The program also generates the following figure (axis labels and legend were added to the plot by using the Plot Editor).



Sample Problem 6-10: Creating a random list

Six singers—John, Mary, Tracy, Mike, Katie, and David—are performing in a competition. Write a MATLAB program that generates a list of a random order in which the singers will perform.

Solution

An integer (1 through 6) is assigned to each name (1 to John, 2 to Mary, 3 to Tracy, 4 to Mike, 5 to Katie, and 6 to David). The program, shown below, first creates a list of the integers 1 through 6 in a random order. The integers are made the elements of six-element vector. This is done by using MATLAB's built-in function `randi` (see Section 3.7) for assigning integers to the elements of the vector. To make sure that all the integers of the elements are different from each other, the integers are assigned one by one. Each integer that is suggested by the `randi` function is compared with all the integers that have been assigned to previous elements. If a match is found, the integer is not assigned, and `randi` is used for suggesting a new integer. Since each singer name is associated with an integer, once the integer list is complete the `switch-case` statement is used to create the corresponding name list.

```
clear, clc
n=6;
```

```
L(1)=randi(n);
for p=2:n
    L(p)=randi(n);
    r=0;
    while r==0
      r=1;
      for k=1:p-1
         if L(k)==L(p)
           L(p)=randi(n);
           r=0;
           break
         end
      end
    end
end
for i=1:n
    switch L(i)
    case 1
        disp('John')
    case 2
        disp('Mary')
    case 3
        disp('Tracy')
    case 4
        disp('Mike')
    case 5
        disp('Katie')
    case 6
        disp('David')
    end
end
```

Assign the first integer to L(1).

Assign the next integer to L(p).

Set r to zero.

See explanation below.

Set r to 1.

for loop compares the integer assigned to L(p) to the integers that have been assigned to previous elements.

If a match if found, a new integer is assigned to L(p) and r is set to zero.

The nested for loop is stopped. The program goes back to the while loop. Since r = 0, the nested loop inside the while loop starts again and checks if the new integer that is assigned to L(p) is equal to an integer that is already in the vector L.

The switch-case statement lists the names according to the values of the integers in the elements of L.

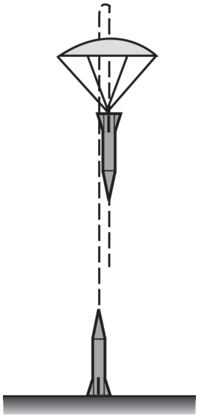
The while loop checks that every new integer (element) that is to be added to the vector L is not equal any of the integers in elements already in the vector L. If a match is found, it keeps generating new integers until the new integer is different from all the integers that are already in x.

When the program is executed, the following is displayed in the Command Window. Obviously, a list in a different order will be displayed every time the program is executed.

```
The performing order is:
```

```
Katie
Tracy
David
Mary
John
Mike
```

Sample Problem 6-11: Flight of a model rocket



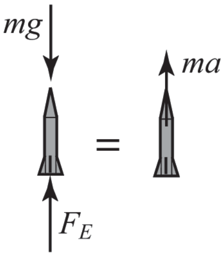
The flight of a model rocket can be modeled as follows. During the first 0.15s the rocket is propelled upward by the rocket engine with a force of 16 N. The rocket then flies up while slowing down under the force of gravity. After it reaches the apex, the rocket starts to fall back down. When its downward velocity reaches 20 m/s, a parachute opens (assumed to open instantly), and the rocket continues to drop at a constant speed of 20 m/s until it hits the ground. Write a program that calculates and plots the speed and altitude of the rocket as a function of time during the flight.

Solution

The rocket is assumed to be a particle that moves along a straight line in the vertical plane. For motion with constant acceleration along a straight line, the velocity and position as a function of time are given by:

$$v(t) = v_0 + at \quad \text{and} \quad s(t) = s_0 + v_0 t + \frac{1}{2}at^2$$

where v_0 and s_0 are the initial velocity and position, respectively. In the computer program the flight of the rocket is divided into three segments. Each segment is calculated in a `while` loop. In every pass the time increases by an increment.



<u>**Segment 1:**</u> The first 0.15s when the rocket engine is on. During this period, the rocket moves up with a constant acceleration. The acceleration is determined by drawing a free body and a mass acceleration diagram (shown on the right). From Newton's second law, the sum of the forces in the vertical direction is equal to the mass times the acceleration (equilibrium equation):

$$+\!\uparrow \Sigma F = F_E - mg = ma$$

Solving the equation for the acceleration gives:

$$a = \frac{F_E - mg}{m}$$

The velocity and height as a function of time are:

$$v(t) = 0 + at \quad \text{and} \quad h(t) = 0 + 0 + \frac{1}{2}at^2$$

where the initial velocity and initial position are both zero. In the computer program this segment starts at $t = 0$, and the looping continues as long as $t < 0.15$ s. The time, velocity, and height at the end of this segment are t_1, v_1, and h_1.

Segment 2: The motion from when the engine stops until the parachute opens. In this segment the rocket moves with a constant deceleration g. The speed and height of the rocket as functions of time are given by:

$$v(t) = v_1 - g(t - t_1) \quad \text{and} \quad h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2$$

In this segment the looping continues until the velocity of the rocket is –20 m/s (negative since the rocket moves down). The time and height at the end of this segment are t_2 and h_2.

Segment 3: The motion from when the parachute opens until the rocket hits the ground. In this segment the rocket moves with constant velocity (zero acceleration). The height as a function of time is given by $h(t) = h_2 - v_{chute}(t - t_2)$, where v_{chute} is the constant velocity after the parachute opens. In this segment the looping continues as long as the height is greater than zero.

A program in a script file that carries out the calculations is shown below.

```
m=0.05;   g=9.81;   tEngine=0.15;   Force=16;   vChute=-20;
Dt=0.01;
clear t v h
n=1;
t(n)=0; v(n)=0; h(n)=0;
% Segment 1
a1=(Force-m*g)/m;
while t(n) < tEngine & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=a1*t(n);
    h(n)=0.5*a1*t(n)^2;
end
v1=v(n); h1=h(n); t1=t(n);
% Segment 2
while v(n) >= vChute & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
```

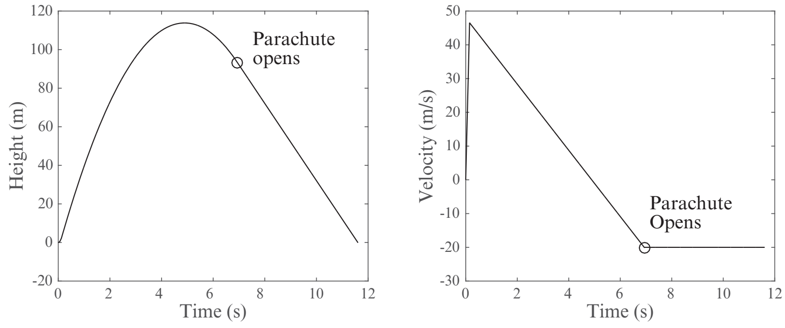
The first `while` loop.

The second `while` loop.

```
    v(n)=v1-g*(t(n)-t1);
    h(n)=h1+v1*(t(n)-t1)-0.5*g*(t(n)-t1)^2;
end
v2=v(n); h2=h(n); t2=t(n);
% Segment 3
while h(n) > 0 & n < 50000
   n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=vChute;
    h(n)=h2+vChute*(t(n)-t2);
end
subplot(1,2,1)
plot(t,h,t2,h2,'o')
subplot(1,2,2)
plot(t,v,t2,v2,'o')
```

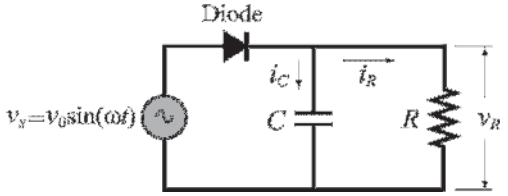
The third `while` loop.

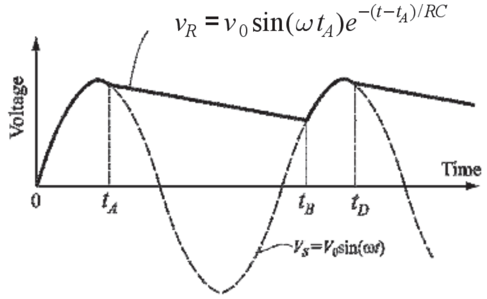
The accuracy of the results depends on the magnitude of the time increment `Dt`. An increment of 0.01 s appears to give good results. The conditional expression in the `while` commands also includes a condition for `n` (if `n` is larger than 50,000 the loop stops). This is done as a precaution to avoid an infinite loop in case there is an error in an of the statements inside the loop. The plots generated by the program are shown below (axis labels and text were added to the plots using the Plot Editor).



Note: The problem can be solved and programmed in different ways. The solution shown here is one option. For example, instead of using `while` loops, the times when the parachute opens and when the rocket hits the ground can be calculated first, and then `for-end` loops can be used instead of the `while` loop. If the times are determined first, it is possible also to use element-by-element calculations instead of loops.

Sample Problem 6-12: AC to DC converter





A half-wave diode rectifier is an electrical circuit that converts AC voltage to DC voltage. A rectifier circuit that consists of an AC voltage source, a diode, a capacitor, and a load (resistor) is shown in the figure. The voltage of the source is $v_s = v_0 \sin(\omega t)$, where $\omega = 2\pi f$, in which f is the frequency. The operation of the circuit is illustrated in the lower diagram where the dashed line shows the source voltage and the solid line shows the voltage across the resistor. In the first cycle, the diode is on (conducting current) from $t = 0$ until $t = t_A$. At this time the diode turns off and the power to the resistor is supplied by the discharging capacitor. At $t = t_B$ the diode turns on again and continues to conduct current until $t = t_D$. The cycle continues as long as the voltage source is on. In this simplified analysis of this circuit, the diode is assumed to be ideal and the capacitor is assumed to have no charge initially (at $t = 0$). When the diode is on, the resistor's voltage and current are given by:

$$v_R = v_0 \sin(\omega t) \quad \text{and} \quad i_R = v_0 \sin(\omega t) / R$$

The current in the capacitor is:

$$i_C = \omega C v_0 \cos(\omega t)$$

When the diode is off, the voltage across the resistor is given by:

$$v_R = v_0 \sin(\omega t_A) e^{-(t-t_A)/RC}$$

The times when the diode switches off (t_A, t_D, and so on) are calculated from the condition $i_R = -i_C$. The diode switches on again when the voltage of the source reaches the voltage across the resistor (time t_B in the figure).

Write a MATLAB program that plots the voltage across the resistor v_R and the voltage of the source v_s as a function of time for $0 \le t \le 70$ ms. The resistance of the load is 1,800 Ω, the voltage source $v_0 = 12$ V, and $f = 60$ Hz. To examine the effect of capacitor size on the voltage across the load, execute the program twice, once with $C = 45$ μF and once with $C = 10$ μF.

Solution

A program that solves the problem is presented below. The program has two parts—one that calculates the voltage v_R when the diode is on, and the other when the diode is off. The `switch` command is used for switching between the two parts. The calculations start with the diode on (the variable `state='on'`), and when $i_R - i_C \leq 0$ the value of `state` is changed to `'off'`, and the program switches to the commands that calculate v_R for this state. These calculations continue until $v_s \geq v_R$, when the program switches back to the equations that are valid when the diode is on.

```
V0=12; C=45e-6; R=1800; f=60;
Tf=70e-3; w=2*pi*f;
clear t VR Vs
t=0:0.05e-3:Tf;
n=length(t);
state='on'
for i=1:n
   Vs(i)=V0*sin(w*t(i));
   switch state
      case 'on'
      VR(i)=Vs(i);
      iR=Vs(i)/R;
      iC=w*C*V0*cos(w*t(i));
      sumI=iR+iC;
      if sumI <= 0
          state='off';
          tA=t(i);
      end
      case 'off'
      VR(i)=V0*sin(w*tA)*exp(-(t(i)-tA)/(R*C));
      if Vs(i) >= VR(i)
         state='on';
      end
   end
end
plot(t,Vs,':',t,VR,'k','linewidth',1)
xlabel('Time (s)'); ylabel('Voltage (V)')
```

Assign `'on'` to the variable `state`.

Calculate the voltage of the source at time t.

Diode is on.

Check if $i_R - i_C \leq 0$.

If true, assign `'off'` to `state`.

Assign a value to t_A.

Diode is off.

Check if $v_s \geq v_R$.

If true, assign `'on'` to the variable `state`.