Definite integrals

This module contains some examples about how one can program MATLAB in order to find a definite integral approximately. Let us concentrate on the following integral:

$$\int_a^b \sin(x^2)dx.$$

We will take $a = 0$ and $b = 5$ to be specific. The explicit techniques of calculus are of no help for evaluating this integral. Therefore, we must use a numerical method to find its value approximately. We will use three different methods: The Midpoint Rule, the Trapezoid Rule and Simpson's Rule to evaluate the same integral.

1 The Midpoint Rule

Let n be a positive integer. In the Midpoint Rule to find $\int_a^b f(x)dx$, one subdivides the interval $[a, b]$ into n equal intervals of width $h = (b - a)/n$. One then selects the midpoints x_i^* of each of these intervals. The value of the integral is approximately

$$h(f(x_1^*) + f(x_2^*) + \ldots + f(x_n^*))$$

In MATLAB, one can make such a computation in essentially two ways: By entering the commands line by line, or by creating an M-file. We will do both. Let us start from a line by line evaluation. Enter the following commands in order in the MATLAB screen:

```
>> a=0;
>> b=5;
>> n=10;
>> h=(b-a)/n;
>> xx=zeros(1,n);
>> for i=1:n
      xx(i)=a+h/2+(i-1)*h;
   end
>> Sum=0;
>> for i=1:n
     Sum=Sum+h*(sin(xx(i)∧2));
   end
>> Sum
```

Here, xx(i) stands for the ith midpoint x_i^*. The command xx=zeros(1,n) allocates space for these variables. The first for loop computes the correct values of the midpoints, and the second for loop computes the sum in the midpoint rule. You should obtain the value 0.2669. You can try to repeat the same sequence with $n = 20$, $n = 100$ and $n = 1000$, which give 0.4966, 0.5269 and 0.5279 respectively. For n greater than 1000, the approximation is always 0.5279, so this is the value of the integral up to 4 decimal digits. The computation with $n = 10$ is very far from the actual value.

Let us now write an M-file for the same purpose. You should click File/New/M-File from the menu to start. We define a function S=sinsquaremid as follows:

```
function S=sinsquaremid(a,b,n)
h=(b-a)/n;
xx=zeros(1,n);
for i=1:n
xx(i)=a+h/2+(i-1)*h;
end
S=0;
for i=1:n
S=S+h*(sin(xx(i)∧2));
end
```

After completion, save the file. Now, on the MATLAB screen, in order to call the function with $a = 0$, $b = 5$ and $n = 10$, write

```
>> sinsquaremid(0,5,10)
```

This should produce the output 0.2669 again. The advantage now is that you can recall the same function easily for different values of n (or a, b), without repeating the same commands over and over. Please try to find the approximation for different values of n.

2 The Trapezoid Rule

There are alternative ways to approximate the same integral. One of these is the trapezoid rule. Instead of approximating the area by rectangles with heights computed from the midpoints, one can form trapezoids with corners $(x_i, 0), (x_i, f(x_i)), (x_{i+1}, f(x_{i+1})$ and $(x_{i+1}, 0)$. Upon adding their contributions, one gets the following approximation for the integral:

$$\frac{h}{2}(f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)).$$

The first and the last coefficients are 1, since the first and the last points belong to only one trapezoid, and all of the intermediate coefficients are 2, since each of those points belong to two trapezoids.

Here is an M-file that computes $\int_a^b \sin(x^2)dx$ using the trapezoid rule with n intervals:

```
function S=sinsquaretrap(a,b,n)
h=(b-a)/n;
x=zeros(1,n+1);
for i=1:n+1
x(i)=a+h*(i-1);
end
S=0;
S=S+(h/2)*(sin(x(1)∧2));
for i=2:n
S=S+h*(sin(x(i)∧2));
end
S=S+(h/2)*(sin(x(n+1)∧2));
end
```

Here, x(i) denotes the point x_{i-1}. We needed this shift since the indexing of variables cannot start from 0 in MATLAB. The two extra lines about S are used for the computation at the first and the last points. The for loop makes the computation at the interior points.

You can call this function by writing

```
>> sinsquaretrap(0,5,10)
```

This produces the value 0.9047. The convergence is not really any better than in the midpoint rule. For $n = 100$ we get 0.5300 and for $n = 1000$ and on we get the correct value 0.5379.

3 Simpson's Rule

Even though the midpoint rule and the trapezoid rule are easy to understand geometrically, and they indeed are useful for computing approximations for the integral, their convergence is rather slow. Making a computation at 1000 points is quite unacceptable for such a simple integral. The problem is, if one has to do repetitive such computations, then the time spent by the computer during the for loops becomes very costly. For this purpose, more efficient methods are developed for numerical integration. Simpson's rule is one of them, and it is not really more difficult to apply than the previous two rules. The advantage, on the other hand, is great.

The idea is to somehow combine the "line fitting" idea of the trapezoid rule, and Lagrange interpolation. Instead of fitting the best line segment to two consecutive points in the trapezoid rule, we can use Lagrange interpolation to fit the best parabola to three consecutive points. We divide the interval into $2n$ segments, and fit a parabola to the consecutive triples $(x_{2k}, f(x_{2k})), (x_{2k+1}, f(x_{2k+1}))$ and $(x_{2k+2}, f(x_{2k+2}))$. Note that the middle point always has odd indices (so we don't take all consecutive triples, just about half of them). Calculating the areas under the parabolic segments gives the approximation:

$$\frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})).$$

(The pattern 142424...241 doesn't have a naive geometric explanation; one really has to integrate the parabolic functions this time to find them.)

Let us write an M-file that computes the approximation by Simpson's rule with $2n$ intervals:

```
function S=sinsquareSimpson(a,b,n)
h=(b-a)/(2*n);
x=zeros(1,2*n+1);
for i=1:2*n+1
x(i)=a+h*(i-1);
end
S=0;
S=S+(h/3)*(sin(x(1)∧2));
for i=1:n-1
S=S+(4*h/3)*(sin(x(2*i)∧2));
```

```
    S=S+(2*h/3)*(sin(x(2*i+1)∧2));
    end
    S=S+(4*h/3)*(sin(x(2*n)∧2));
    S=S+(h/3)*(sin(x(2*n+1)∧2));
    end
```

This time, computation with $n = 10$ gives the value 0.4795, with $n = 20$ we get 0.5263 and already with $n = 50$ we get the correct value 0.5279. So the Simpson's rule gives the correct value for much smaller n compared to the Midpoint and the Trapezoid rules. This would save a lot of time in any serious application.

Linear Algebra

1 Gaussian Elimination with Partial Pivoting

Solve the following set of algebraic equations:

$$\begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{-1} & \mathbf{4} \\ \mathbf{4} & \mathbf{2} & \mathbf{-5} & \mathbf{-7} \\ \mathbf{1} & \mathbf{2} & \mathbf{5} & \mathbf{7} \\ \mathbf{4} & \mathbf{2} & \mathbf{4} & \mathbf{7} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 18 \\ -35 \\ 48 \\ 48 \end{bmatrix}$$

```
>> A=[1 2 -1 4 18; 4 2 -5 -7 -35; 1 2 5 7 48; 4 2 4 7 48]

A =

     1     2    -1     4    18
     4     2    -5    -7   -35
     1     2     5     7    48
     4     2     4     7    48

>> A=A([2 1 3 4],:)

A =

     4     2    -5    -7   -35
     1     2    -1     4    18
     1     2     5     7    48
     4     2     4     7    48

>> A(2,:)=A(2,:)-A(2,1)/A(1,1)*A(1,:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
    1.0000    2.0000    5.0000    7.0000   48.0000
    4.0000    2.0000    4.0000    7.0000   48.0000

>> A(3,:)=A(3,:)-A(3,1)/A(1,1)*A(1,:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
         0    1.5000    6.2500    8.7500   56.7500
    4.0000    2.0000    4.0000    7.0000   48.0000
```

```
>> A(4,:)=A(4,:)-A(4,1)/A(1,1)*A(1,:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
         0    1.5000    6.2500    8.7500   56.7500
         0         0    9.0000   14.0000   83.0000

>> A(3,:)=A(3,:)-A(3,2)/A(2,2)*A(2,:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
         0         0    6.0000    3.0000   30.0000
         0         0    9.0000   14.0000   83.0000

>> A=A([1 2 4 3],:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
         0         0    9.0000   14.0000   83.0000
         0         0    6.0000    3.0000   30.0000

>> A(4,:)=A(4,:)-A(4,3)/A(3,3)*A(3,:)

A =

    4.0000    2.0000   -5.0000   -7.0000  -35.0000
         0    1.5000    0.2500    5.7500   26.7500
         0         0    9.0000   14.0000   83.0000
         0         0         0   -6.3333  -25.3333

>> t=A(4,5)/A(4,4)

t =

     4

>> z=(A(3,5)-A(3,4)*t)/A(3,3)

z =

     3
```

```
>> y=(A(2,5)-A(2,4)*t-A(2,3)*z)/A(2,2)

y =

     2

>> y=(A(1,5)-A(1,4)*t-A(1,3)*z-A(1,2)*y)/A(1,1)

y =

     1

>>
```

2 Matrix Inverse

Find the inverse of the following matrix:

$$\begin{bmatrix} 1 & 2 & -1 & 4 \\ 4 & 2 & -5 & -7 \\ 1 & 2 & 5 & 7 \\ 4 & 2 & 4 & 7 \end{bmatrix}$$

```
>> B=[1 2 -1 4 1 0 0 0; 4 2 -5 -7 0 1 0 0; 1 2 5 7 0 0 1 0; 4 2 4 7 0 0 0 1]

B =

     1     2    -1     4     1     0     0     0
     4     2    -5    -7     0     1     0     0
     1     2     5     7     0     0     1     0
     4     2     4     7     0     0     0     1

>> B(2,:)=B(2,:)-B(2,1)/B(1,1)*B(1,:)

B =

     1     2    -1     4     1     0     0     0
     0    -6    -1   -23    -4     1     0     0
     1     2     5     7     0     0     1     0
     4     2     4     7     0     0     0     1

>> B(3,:)=B(3,:)-B(3,1)/B(1,1)*B(1,:)

B =
     1     2    -1     4      1     0     0     0
     0    -6    -1   -23     -4     1     0     0
     0     0     6     3     -1     0     1     0
     4     2     4     7      0     0     0     1

>> B(4,:)=B(4,:)-B(4,1)/B(1,1)*B(1,:)

B =

     1     2    -1     4     1     0     0     0
     0    -6    -1   -23    -4     1     0     0
     0     0     6     3    -1     0     1     0
     0    -6     8    -9    -4     0     0     1

>> B(3,:)=B(3,:)-B(3,2)/B(2,2)*B(2,:)

B =

     1     2    -1     4      1     0     0     0
     0    -6    -1   -23     -4     1     0     0
     0     0     6     3     -1     0     1     0
     0    -6     8    -9     -4     0     0     1

>> B(4,:)=B(4,:)-B(4,2)/B(2,2)*B(2,:)

B =

     1     2    -1     4      1     0     0     0
     0    -6    -1   -23     -4     1     0     0
     0     0     6     3     -1     0     1     0
     0     0     9    14      0    -1     0     1

>> B(4,:)=B(4,:)-B(4,3)/B(3,3)*B(3,:)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
         0   -6.0000   -1.0000  -23.0000   -4.0000    1.0000         0         0
         0         0    6.0000    3.0000   -1.0000         0    1.0000         0
         0         0         0    9.5000    1.5000   -1.0000   -1.5000    1.0000

>> B(2,:)=B(2,:)/B(2,2)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
         0    1.0000    0.1667    3.8333    0.6667   -0.1667         0         0
    0         0         6.0000    3.0000   -1.0000         0    1.0000         0
    0         0         0         9.5000    1.5000   -1.0000   -1.5000    1.0000

>> B(3,:)=B(3,:)/B(3,3)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
    0         1.0000    0.1667    3.8333    0.6667   -0.1667         0         0
    0         0         1.0000    0.5000   -0.1667         0    0.1667         0
    0         0         0         9.5000    1.5000   -1.0000   -1.5000    1.0000

>> B(4,:)=B(4,:)/B(4,4)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
    0         1.0000    0.1667    3.8333    0.6667   -0.1667         0         0
    0         0         1.0000    0.5000   -0.1667         0    0.1667         0
    0         0         0         1.0000    0.1579   -0.1053   -0.1579    0.1053

>> B(3,:)=B(3,:)-B(3,4)*B(4,:)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
    0         1.0000    0.1667    3.8333    0.6667   -0.1667         0         0
    0         0         1.0000         0   -0.2456    0.0526    0.2456   -0.0526
    0         0         0         1.0000    0.1579   -0.1053   -0.1579    0.1053

>> B(2,:)=B(2,:)-B(2,4)*B(4,:)

B =

    1.0000    2.0000   -1.0000    4.0000    1.0000         0         0         0
    0         1.0000    0.1667         0    0.0614    0.2368    0.6053   -0.4035
    0         0         1.0000         0   -0.2456    0.0526    0.2456   -0.0526
    0         0         0         1.0000    0.1579   -0.1053   -0.1579    0.1053

>> B(1,:)=B(1,:)-B(1,4)*B(4,:)

B =

    1.0000    2.0000   -1.0000         0    0.3684    0.4211    0.6316   -0.4211
    0         1.0000    0.1667         0    0.0614    0.2368    0.6053   -0.4035
    0         0         1.0000         0   -0.2456    0.0526    0.2456   -0.0526
    0         0         0         1.0000    0.1579   -0.1053   -0.1579    0.1053
```

```
>> B(2,:)=B(2,:)-B(2,3)*B(3,:)

B =

   1.0000    2.0000   -1.0000   0           0.3684    0.4211    0.6316   -0.4211
   0         1.0000    0         0           0.1023    0.2281    0.5643   -0.3947
   0         0         1.0000    0          -0.2456    0.0526    0.2456   -0.0526
   0         0         0         1.0000      0.1579   -0.1053   -0.1579    0.1053

>> B(1,:)=B(1,:)-B(1,3)*B(3,:)

B =

   1.0000    2.0000         0         0          0.1228    0.4737    0.8772   -0.4737
   0         1.0000         0         0          0.1023    0.2281    0.5643   -0.3947
   0         0              1.0000    0         -0.2456    0.0526    0.2456   -0.0526
   0         0              0         1.0000     0.1579   -0.1053   -0.1579    0.1053

>> B(1,:)=B(1,:)-B(1,2)*B(2,:)

B =

   1.0000  0         0         0          -0.0819    0.0175   -0.2515    0.3158
   0        1.0000  0         0           0.1023    0.2281    0.5643   -0.3947
   0        0         1.0000  0          -0.2456    0.0526    0.2456   -0.0526
   0        0         0         1.0000     0.1579   -0.1053   -0.1579    0.1053
```

3 Power Method for Eigenvalues

One dimensional wave equation for elastic deformations in a bar (fixed at both ends) can be written as

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$

where c is the sound velocity. The boundary conditions are

$$u(0,t) = 0$$

$$u(1,t) = 0$$

Assuming a solution in the form of $u = ye^{i\omega t}$, the governing equations take the form

$$\frac{\partial^2 y}{\partial x^2} + \frac{\omega^2}{c^2} y = 0$$

This equation can be discretized by using central differences as follows:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} + \frac{\omega^2}{c^2} y_i = 0$$

Find the largest eigenvalue and the corresponding frequency. Divide the computational domain to four intervals ($\Delta x = 0.25$ m**)**. Take c=2000 m/s.

```
>> A=zeros(3,3)

A =

     0     0     0
     0     0     0
     0     0     0

A = [32 -16 0; -16 32 -16;0 -16 32]                          (32=2/DX^2)

    32   -16     0
   -16    32   -16
     0   -16    32

>> x=[1 1 1]'   ;                       (initial guess)

>> y=A*x ;                                (Step 1)

>> m0=x'*x ;

>> m1=x'*y;

>> m2=y'*y;

>> q=m1/m0

q =

   10.6667

>> d=sqrt(m2/m0-q^2);

>> x=y/sqrt(y(1)^2+y(2)^2+y(3)^2);
```

Goto Step 1 and proceed until q converges.

The largest eigenvalue is found to be 54.627.
The corresponding eigenvector is 0.5, -0.707,0.5.

Newton-Raphson Method for Nonlinear Equations

The frequency that corresponds to the largest eigenvalue is 14782 Hz. Find two positive roots of the following nonlinear equation using Newton-Raphson method:

$$\cos(x) - x^2 \sin(x) = 0$$

```
x =

     1

>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    0.9017

>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    0.8952

>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    0.8952


x =

     3

>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    3.2853

>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    3.2380
```

```
>> x=x-(cos(x)-x^2*sin(x))/(-sin(x)-(2*x*sin(x)+x^2*cos(x)))

x =

    3.2368
```