MAT1011 – Calculus for Engineers (MATLAB), Fall Semester 2020-2021

Digital Assignment SL. 7, Experiment – 4A: Double Integrals and Change of Order of Integration

By: Jonathan Rufus Samuel (20BCT0332) Date: 17.12.2020

Q1) Write a program for the double integral ∬R (x - 3y^2) dA where R = {(x,y) | 0<=x<=2 , 1<=y<=2} and visualize it.

A: Code is as follows:

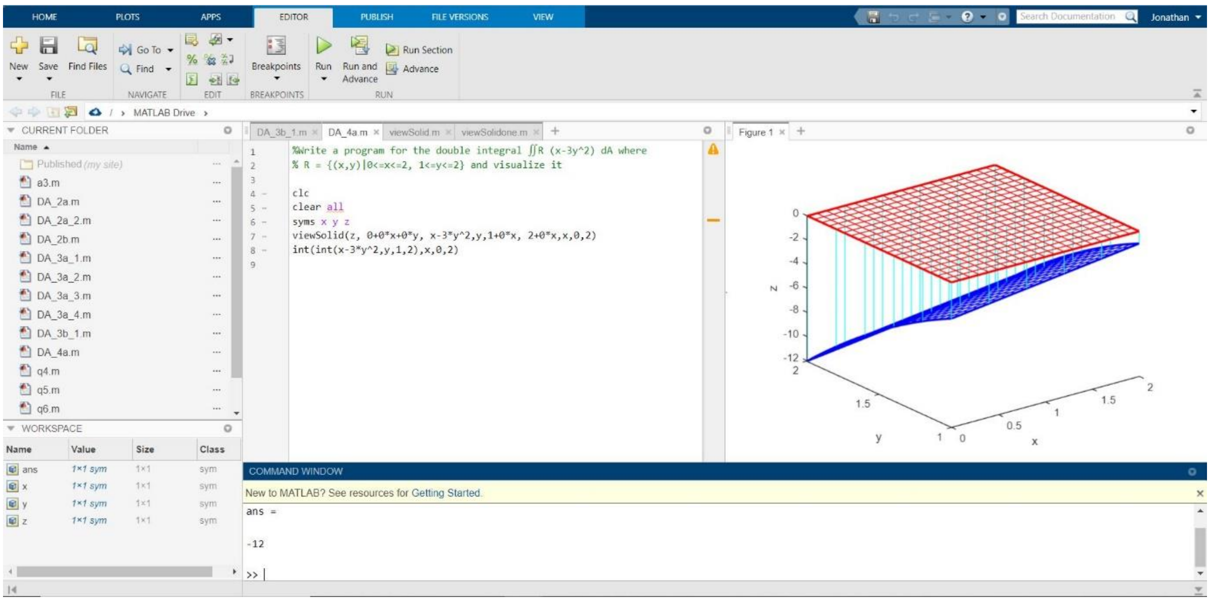
```
%Write a program for the double integral ∬R (x-3y^2) dA where
% R = {(x,y)|0<=x<=2, 1<=y<=2} and visualize it


clc
clear all
syms x y z
viewSolid(z, 0+0*x+0*y, x-3*y^2,y,1+0*x, 2+0*x,x,0,2)
int(int(x-3*y^2,y,1,2),x,0,2)
```

Output (via Command Window):

```
ans =

-12
```



Additional Files (To be Added to MatLab File Directory):

1) viewSolid.m

```
function viewSolid(zvar, F, G, yvar, f, g, xvar, a, b)
%VIEWSOLID is a version for MATLAB of the routine on page 161
% of "Multivariable Calculus and Mathematica" for viewing the region
% bounded by two surfaces for the purpose of setting up triple integrals.
% The arguments are entered from the inside out.
% There are two forms of the command --- either f, g,
% F, and G can be vectorized functions, or else they can
% be symbolic expressions. xvar, yvar, and zvar can be
% either symbolic variables or strings.
% The variable xvar (x, for example) is on the
% OUTSIDE of the triple integral, and goes between CONSTANT limits a and b.
% The variable yvar goes in the MIDDLE of the triple integral, and goes
% between limits which must be expressions in one variable [xvar].
% The variable zvar goes in the INSIDE of the triple integral, and goes
% between limits which must be expressions in two
% variables [xvar and yvar]. The lower surface is plotted in red, the
% upper one in blue, and the "hatching" in cyan.
%
% Examples: viewSolid(z, 0, (x+y)/4, y, x/2, x, x, 1, 2)
% gives the picture on page 163 of "Multivariable Calculus and Mathematica"
% and the picture on page 164 of "Multivariable Calculus and Mathematica"
% can be produced by
% viewSolid(z, x^2+3*y^2, 4-y^2, y, -sqrt(4-x^2)/2, sqrt(4-x^2)/2, ...
% x, -2, 2,)
% One can also type viewSolid('z', @(x,y) 0, ...
% @(x,y)(x+y)/4, 'y', @(x) x/2, @(x) x, 'x', 1, 2)
%


if isa(f, 'sym') % case of symbolic input
ffun=inline(vectorize(f+0*xvar),char(xvar));
gfun=inline(vectorize(g+0*xvar),char(xvar));
Ffun=inline(vectorize(F+0*xvar),char(xvar),char(yvar));
Gfun=inline(vectorize(G+0*xvar),char(xvar),char(yvar));
oldviewSolid(char(xvar), double(a), double(b), ...
char(yvar), ffun, gfun, char(zvar), Ffun, Gfun)
else
oldviewSolid(char(xvar), double(a), double(b), ...
char(yvar), f, g, char(zvar), F, G)
end
%%%%%%% subfunction goes here %%%%%%
function oldviewSolid(xvar, a, b, yvar, f, g, zvar, F, G)
for counter=0:20
  xx = a + (counter/20)*(b-a);
  YY = f(xx)*ones(1, 21)+((g(xx)-f(xx))/20)*(0:20);
  XX = xx*ones(1, 21);
%% The next lines inserted to make bounding curves thicker.
  widthpar=0.5;
  if counter==0, widthpar=2; end
  if counter==20, widthpar=2; end
%% Plot curves of constant x on surface patches.
 plot3(XX, YY, F(XX, YY).*ones(1,21), 'r', 'LineWidth', widthpar);
 hold on
```

```
 plot3(XX, YY, G(XX, YY).*ones(1,21), 'b', 'LineWidth', widthpar);
end;
%% Now do the same thing in the other direction.
XX = a*ones(1, 21)+((b-a)/20)*(0:20);
%% Normalize sizes of vectors.
YY=0:2; ZZ1=0:20; ZZ2=0:20;
for counter=0:20,
%% The next lines inserted to make bounding curves thicker.
  widthpar=0.5;
  if counter==0, widthpar=2; end
  if counter==20, widthpar=2; end
    for i=1:21,
       YY(i)=f(XX(i))+(counter/20)*(g(XX(i))-f(XX(i)));
       ZZ1(i)=F(XX(i),YY(i));
       ZZ2(i)=G(XX(i),YY(i));
    end;
  plot3(XX, YY, ZZ1, 'r', 'LineWidth',widthpar);
  plot3(XX, YY, ZZ2, 'b', 'LineWidth',widthpar);
end;
%% Now plot vertical lines.
for u = 0:0.2:1,
  for v = 0:0.2:1,
   x=a + (b-a)*u; y = f(a + (b-a)*u) +(g(a + (b-a)*u)-f(a + (b-a)*u))*v;
   plot3([x, x], [y, y], [F(x,y), G(x, y)], 'c');
  end;
end;
xlabel(xvar)
ylabel(yvar)
zlabel(zvar)
hold off
```

2) viewSolidone.m

```
function viewSolidone(zvar, F, G, xvar, f, g, yvar, a, b)
%VIEWSOLID is a version for MATLAB of the routine on page 161
%  of "Multivariable Calculus and Mathematica" for viewing the region
%  bounded by two surfaces for the purpose of setting up triple integrals.
%  The arguments are entered from the inside out.
%  There are two forms of the command --- either f, g,
%  F, and G can be vectorized functions, or else they can
%  be symbolic expressions. xvar, yvar, and zvar can be
%  either symbolic variables or strings.
%  The variable xvar (x, for example) is on the
%  OUTSIDE of the triple integral, and goes between CONSTANT limits a and b.
%  The variable yvar goes in the MIDDLE of the triple integral, and goes
%  between limits which must be expressions in one variable [xvar].
%  The variable zvar goes in the INSIDE of the triple integral, and goes
%  between limits which must be expressions in two
%  variables [xvar and yvar].  The lower surface is plotted in red, the
%  upper one in blue, and the "hatching" in cyan.
%
% Examples: viewSolid(z, 0, (x+y)/4, y, x/2, x, x, 1, 2)
% gives the picture on page 163 of "Multivariable Calculus and Mathematica"
% and the picture on page 164 of "Multivariable Calculus and Mathematica"
% can be produced by
%     viewSolid(z, x^2+3*y^2, 4-y^2, y, -sqrt(4-x^2)/2, sqrt(4-x^2)/2, ...
```

```
%               x, -2, 2,)
% One can also type viewSolid('z', @(x,y) 0, ...
% @(x,y)(x+y)/4, 'y', @(x) x/2, @(x) x, 'x', 1, 2)
%


if isa(f, 'sym') % case of symbolic input
    ffun=inline(vectorize(f+0*yvar),char(yvar));
    gfun=inline(vectorize(g+0*yvar),char(yvar));
    Ffun=inline(vectorize(F+0*xvar),char(xvar),char(yvar));
    Gfun=inline(vectorize(G+0*xvar),char(xvar),char(yvar));
    oldviewSolid(char(yvar),double(a), double(b), ...
       char(xvar), ffun, gfun, char(zvar), Ffun, Gfun)
else
   oldviewSolid(char(yvar),double(a),double(b),char(xvar), f, g, char(zvar), F, G)
end
%%%%%%% subfunction goes here %%%%%%
function oldviewSolid(yvar,a , b, xvar, f, g, zvar, F, G)
for counter=0:30
  yy= a + (counter/30)*(b-a);
  XX = f(yy)*ones(1, 31)+((g(yy)-f(yy))/30)*(0:30);
  YY = yy*ones(1, 31);
%% The next lines inserted to make bounding curves thicker.
widthpar=0.5;
if counter==0, widthpar=2; end
if counter==20, widthpar=2; end
%% Plot curves of constant x on surface patches.
plot3(YY,XX, F(XX, YY).*ones(1,31), 'r', 'LineWidth', widthpar);
hold on
plot3(YY,XX, G(XX, YY).*ones(1,31), 'b', 'LineWidth', widthpar);
end;
%% Now do the same thing in the other direction.
YY = a*ones(1, 31)+((b-a)/30)*(0:30);
%% Normalize sizes of vectors.
XX=0:2; ZZ1=0:30; ZZ2=0:30;
for counter=0:30,
%% The next lines inserted to make bounding curves thicker.
widthpar=0.5;
if counter==0, widthpar=2; end
if counter==30, widthpar=2; end
for i=1:31,
XX(i)=f(YY(i))+(counter/30)*(g(YY(i))-f(YY(i)));
ZZ1(i)=F(YY(i),XX(i));
ZZ2(i)=G(YY(i),XX(i));
end;
plot3(YY,XX, ZZ1, 'r', 'LineWidth',widthpar);
plot3(YY,XX, ZZ2, 'g', 'LineWidth',widthpar);
end;
%% Now plot vertical lines.
for u = 0:0.09:1,
for v = 0:0.09:1,
y=a + (b-a)*u; x = f(a + (b-a)*u) +(g(a + (b-a)*u)-f(a + (b-a)*u))*v;
plot3([y, y], [x, x], [F(x,y), G(x, y)], 'c');
end;
end;
xlabel(xvar)
```

```
ylabel(yvar)
zlabel(zvar)
hold off
```
