Aufgabe 1: Polynome

1. ***Simple*** implementation:

Listing 1: Polynom Common

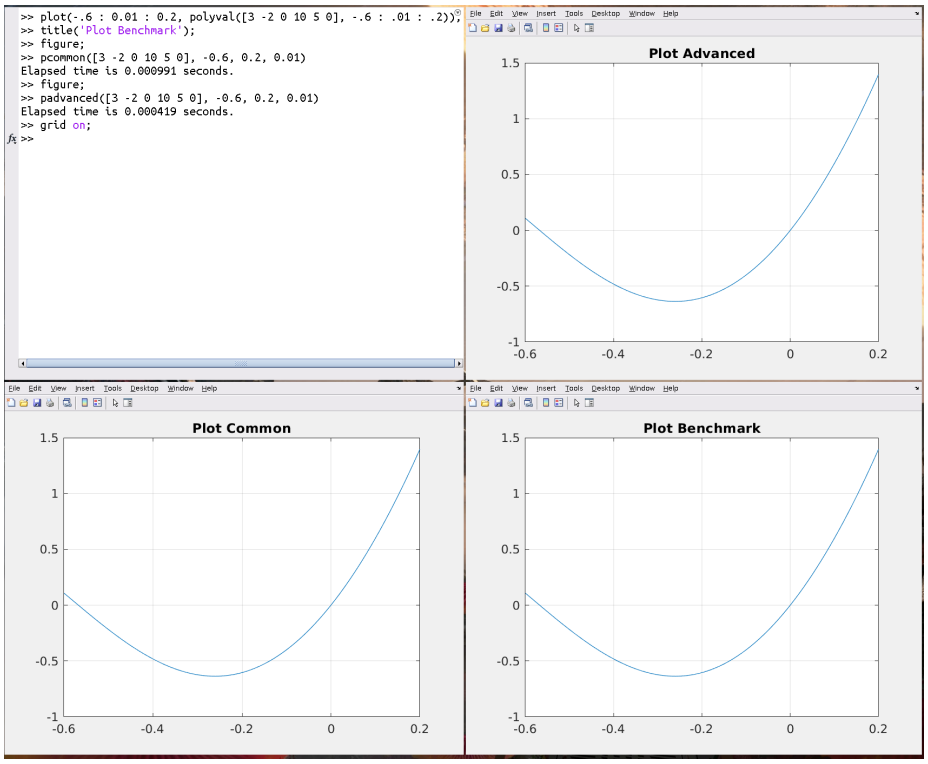
```
function pcommon(A, a, b, h)

    if nargin == 3
        h = 0.01;
    end

    tic

    N = (b - a) / h + 1;

    Y = [];

    x_ = 0;

    for i = a : h : b
        x_ = x_ + 1;
        p = 0;
        for x = (size(A, 2) - 1) : -1 : 0
            p = p + A(size(A, 2) - x) * i ^ x;
        end
        Y = [Y p];
    end

    toc

    plot(a : h : b, Y);

    title('Plot Common');

    grid on;

end
```

2. ***Smart*** implementation:

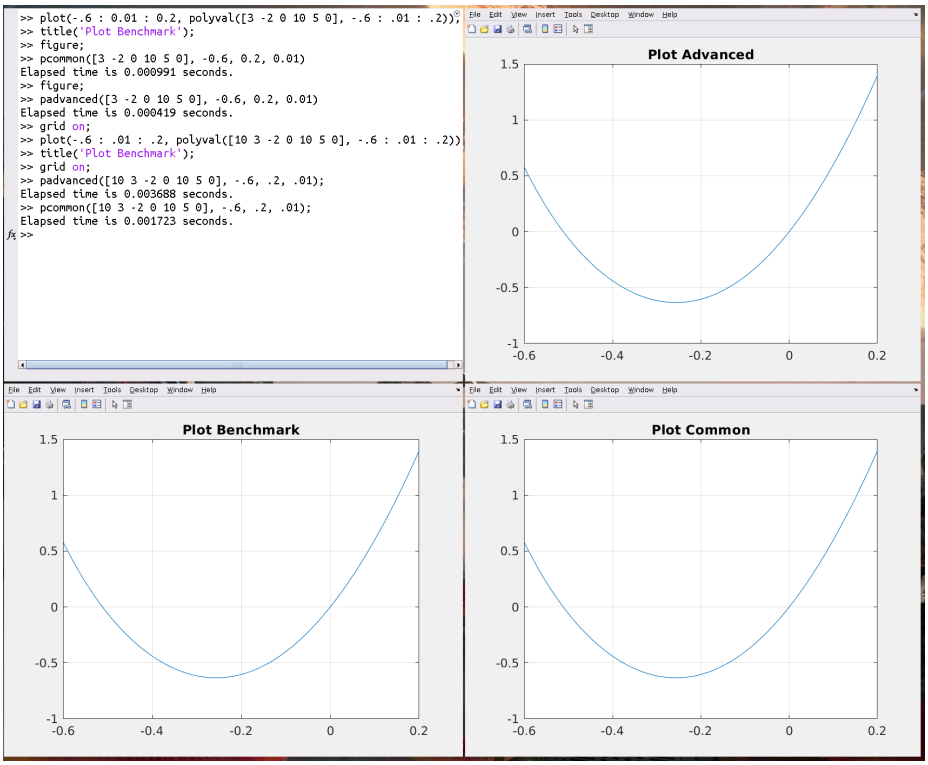
Listing 2: Polynom Advanced

```
function padvanced(A, a, b, h)

    if nargin == 3
        h = 0.01;
    end

    tic

    % to the power vector
    p1 = size(A,2) - 1 : -1 : 0;

    % needed to transform Y
    p2 = ones(size(A))';

    % x values
    Y1 = (a : h : b) .* p2;

    % power every value
    Y2 = Y1' .^ p1;

    % mulitply every value with input array
    Y3 = Y2 .* A;

    % calculate final p(x)
    Y4 = sum(Y3, 2);

    toc

    plot(a : h : b, Y4);

    title('Plot Advanced');

    grid on;

end
```

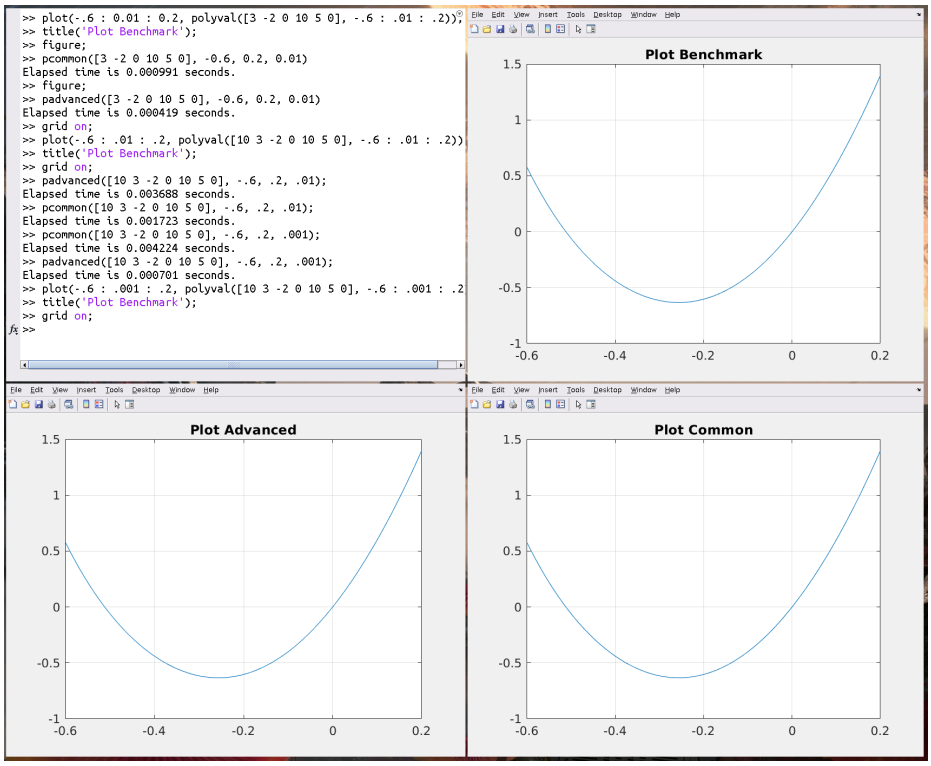
3. Run 1


>> plot(-.6 : 0.01 : 0.2, polyval([3 -2 0 10 5 0], -.6 : .01 : .2));
>> title('Plot Benchmark');
>> figure;
>> pcommon([3 -2 0 10 5 0], -0.6, 0.2, 0.01)
Elapsed time is 0.000991 seconds.
>> figure;
>> padvanced([3 -2 0 10 5 0], -0.6, 0.2, 0.01)
Elapsed time is 0.000419 seconds.
>> grid on;
>>
Plot Advanced
Plot Common
Plot Benchmark

4. Run 2


>> plot(-.6 : 0.01 : 0.2, polyval([3 -2 0 10 5 0], -.6 : .01 : .2));
>> title('Plot Benchmark');
>> figure;
>> pcommon([3 -2 0 10 5 0], -0.6, 0.2, 0.01)
Elapsed time is 0.000991 seconds.
>> figure;
>> padvanced([3 -2 0 10 5 0], -0.6, 0.2, 0.01)
Elapsed time is 0.000419 seconds.
>> grid on;
>> plot(-.6 : .01 : .2, polyval([10 3 -2 0 10 5 0], -.6 : .01 : .2))
>> title('Plot Benchmark');
>> grid on;
>> padvanced([10 3 -2 0 10 5 0], -.6, .2, .01);
Elapsed time is 0.003688 seconds.
>> pcommon([10 3 -2 0 10 5 0], -.6, .2, .01);
Elapsed time is 0.001723 seconds.
>>
Plot Advanced
Plot Benchmark
Plot Common

5. Run 3



6. **Comparison:**

   The advanced implementation is about 5 times faster than the simple one in the above three runs.

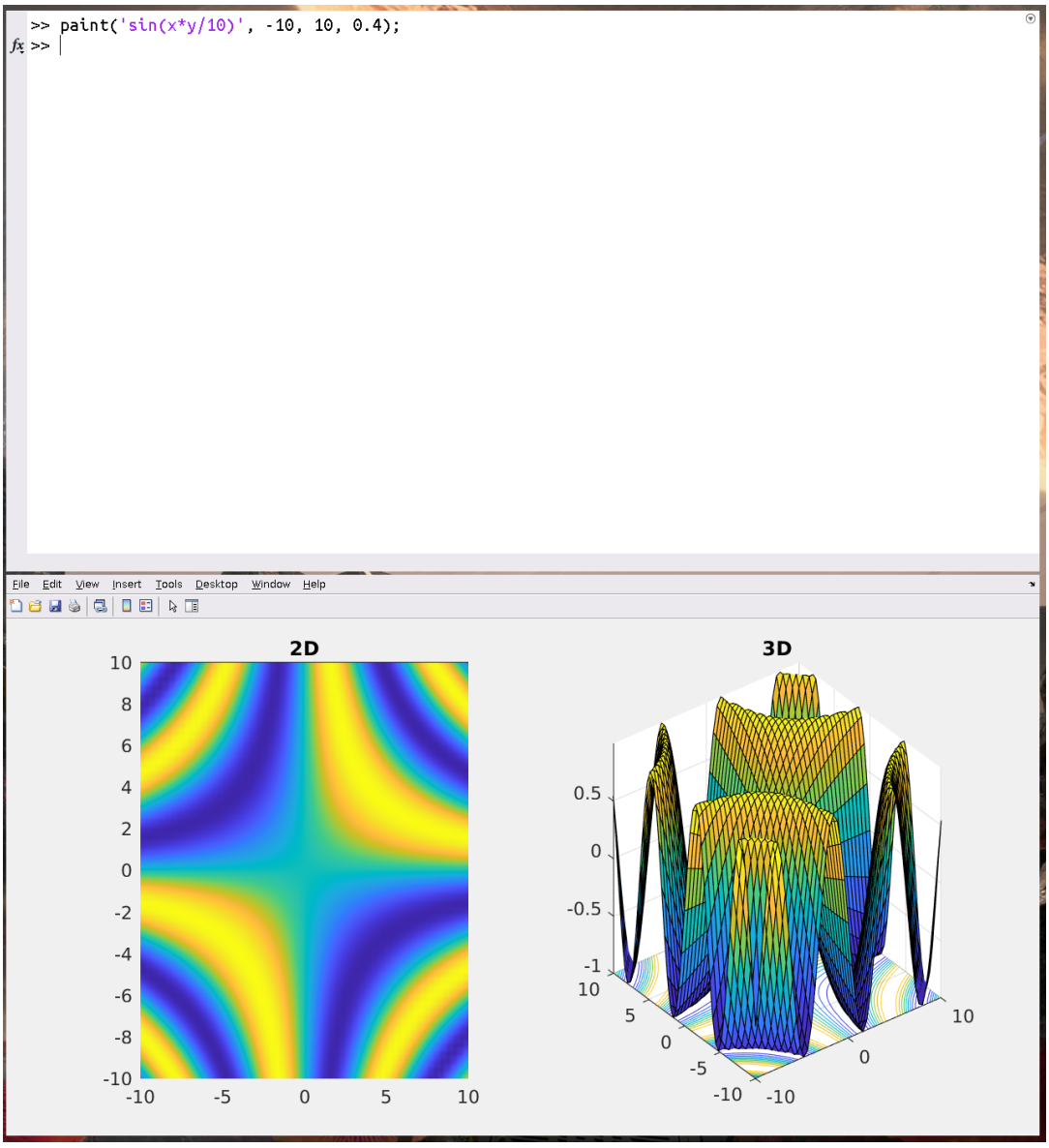
Aufgabe 2: Malen nach Zahlen (Pt 1)

1. Source Code:

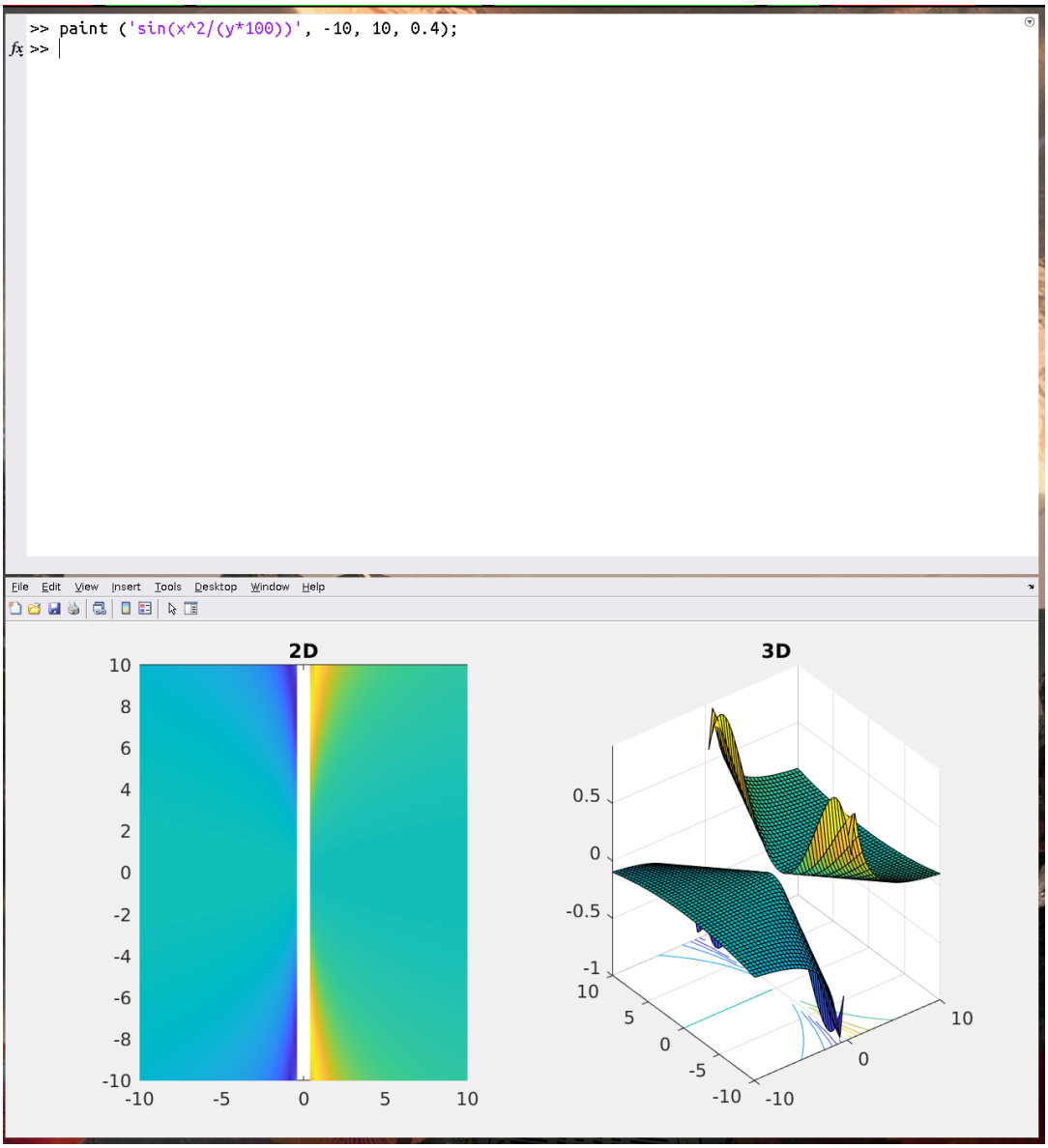
Listing 3: paint.m

```
function paint(fnstr, a, b, h)

    if nargin == 3
        h = 0.2;
    end

    N = (b - a) / h + 1;

    Z = zeros(N);

    x_ = 0;

    for x = a : h : b
        x_ = x_ + 1;
        y_ = 0;
        for y = a : h : b
            y_ = y_ + 1;
            Z(x_, y_) = eval(fnstr);
        end
    end

    tiledlayout(1, 2);
    nexttile;
    pcolor(a : h : b, a : h : b, Z);
    shading interp;
    title(2D);
    nexttile;
    surfc(a : h : b, a : h : b, Z);
    title(3D);


end
```

2. Run 1


>> paint('sin(x*y/10)', -10, 10, 0.4);
>>
File Edit View Insert Tools Desktop Window Help
2D
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10 -5 0 5 10
3D
0.5
0
-0.5
-1
10
5
0
-5
-10
-10
0
10

3. Run 2


>> paint ('sin(x^2/(y*100))', -10, 10, 0.4);
>>
File Edit View Insert Tools Desktop Window Help
2D
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10 -5 0 5 10
3D
0.5
0
-0.5
-1
10
5
0
-5
-10
-10
0
10

Aufgabe 3: Malen nach Zahlen (Pt 2)

1. Source Code:

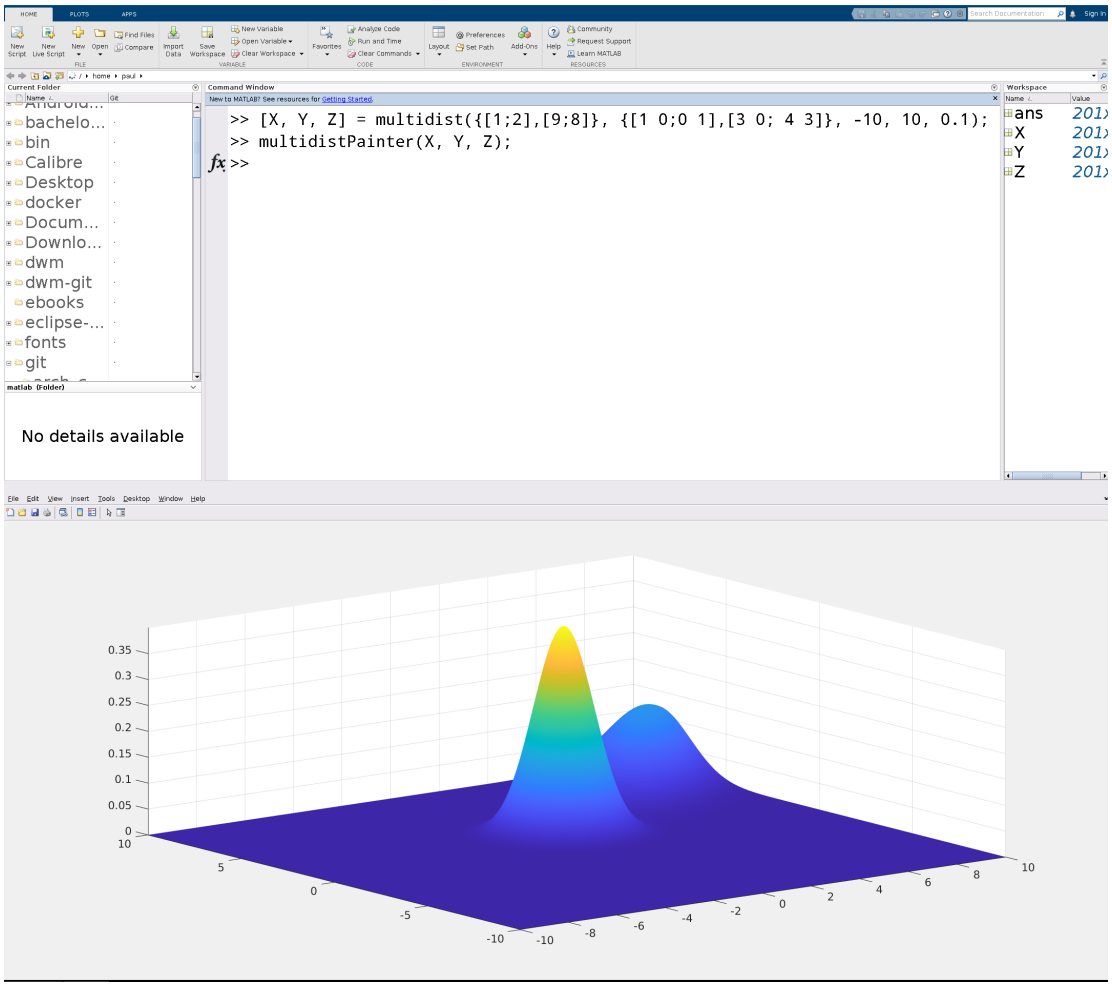
Listing 4: multidist.m

```
function [Xout, Yout, Zout] = multidist(mu, sigma, a, b, h)

    if nargin == 4
        h = 0.01;
    end

    N = (b - a) / h + 1;
    x = zeros(N);
    y = zeros(N);
    z = zeros(N);
    xc = 0;

    for x_ = a : h : b
        yc = 0;
        xc = xc + 1;
        for y_ = a : h : b
            yc = yc + 1;
            x(xc, yc) = x_;
            y(xc, yc) = y_;
            X = [x_; y_];
            z(xc, yc) = 0;
            for i = 1 : 1 : size(mu, 2)
                p1 = 1 / ((2 * pi)^(size(mu{i}, 2) / 2) * det(sigma{i})
                    ^0.5);
                p2 = exp(-0.5*(X - mu{i})' * inv(sigma{i}) * (X - mu{i}))
                    ;
                p_x = p1 * p2;
                z(xc, yc) = z(xc, yc) + p_x;
            end
        end
    end

    Xout = x;
    Yout = y;
    Zout = z;

end
```

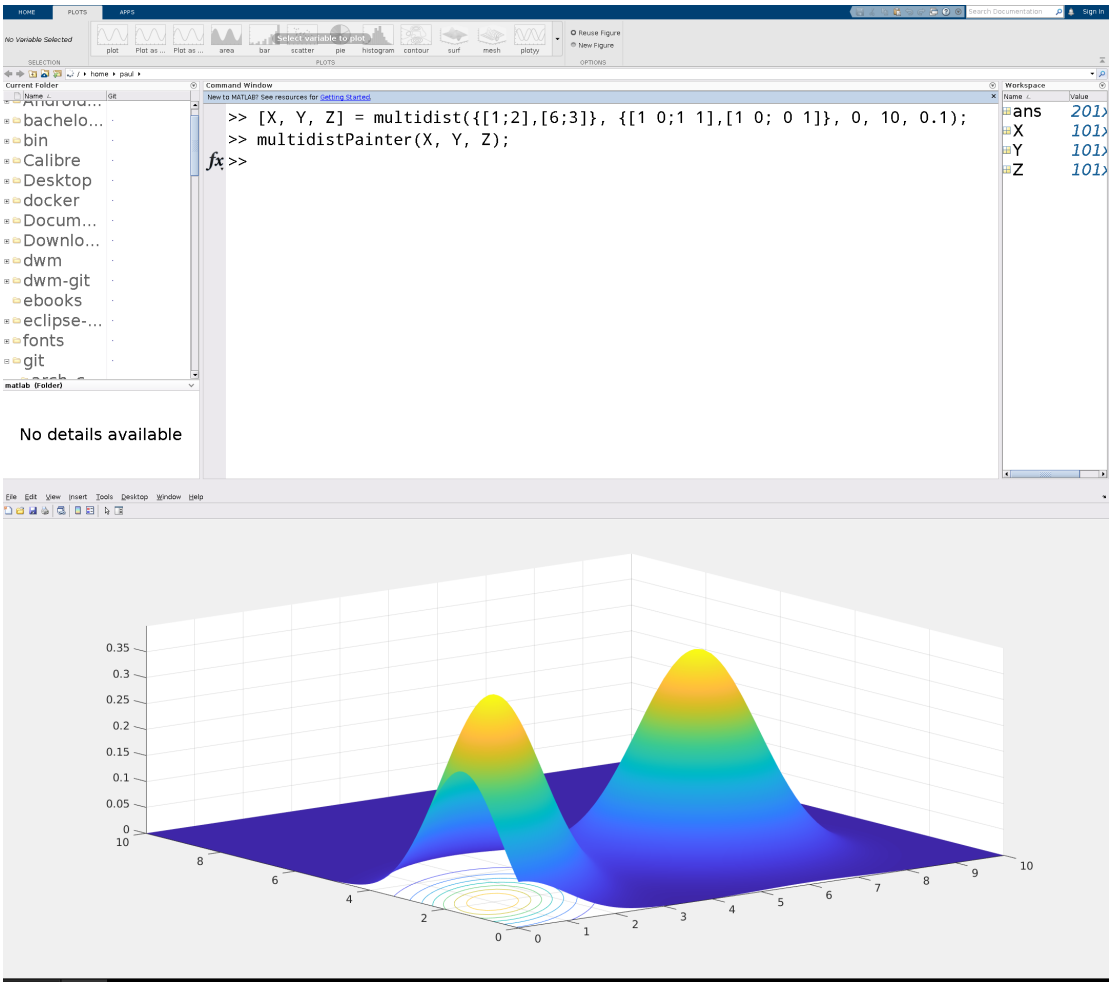
Listing 5: multidistPainter.m

```
function multidistPainter(X, Y, Z)
    surfc(X, Y, Z);
    shading interp;
end
```

2. Run 1


>> [X, Y, Z] = multidist({[1;2],[9;8]}, {[1 0;0 1],[3 0; 4 3]}, -10, 10, 0.1);
>> multidistPainter(X, Y, Z);
>>
No details available
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10

3. Run 2


>> [X, Y, Z] = multidist({[1;2],[6;3]}, {[1 0;1 1],[1 0; 0 1]}, 0, 10, 0.1);
>> multidistPainter(X, Y, Z);
>>
No details available
ans 201x
X 101x
Y 101x
Z 101x
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0