

Aufgabe 1: Polynome

1. *Simple* implementation:

Listing 1: Polynom Common

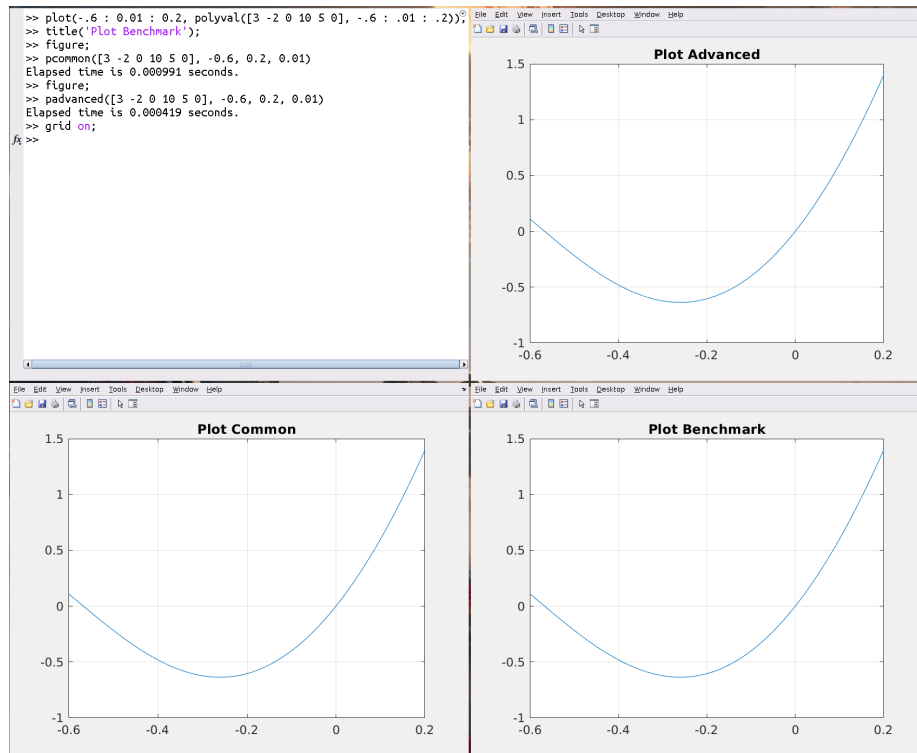
```
1 function pcommon(A, a, b, h)
2
3     if nargin == 3
4         h = 0.01;
5     end
6
7     tic
8
9     N = (b - a) / h + 1;
10
11     Y = [];
12
13     x_ = 0;
14
15     for i = a : h : b
16         x_ = x_ + 1;
17         p = 0;
18         for x = (size(A, 2) - 1) : -1 : 0
19             p = p + A(size(A, 2) - x) * i ^ x;
20         end
21         Y = [Y p];
22     end
23
24     toc
25
26     plot(a : h : b, Y);
27
28     title('Plot Common');
29
30     grid on;
31
32 end
```

2. *Smart* implementation:

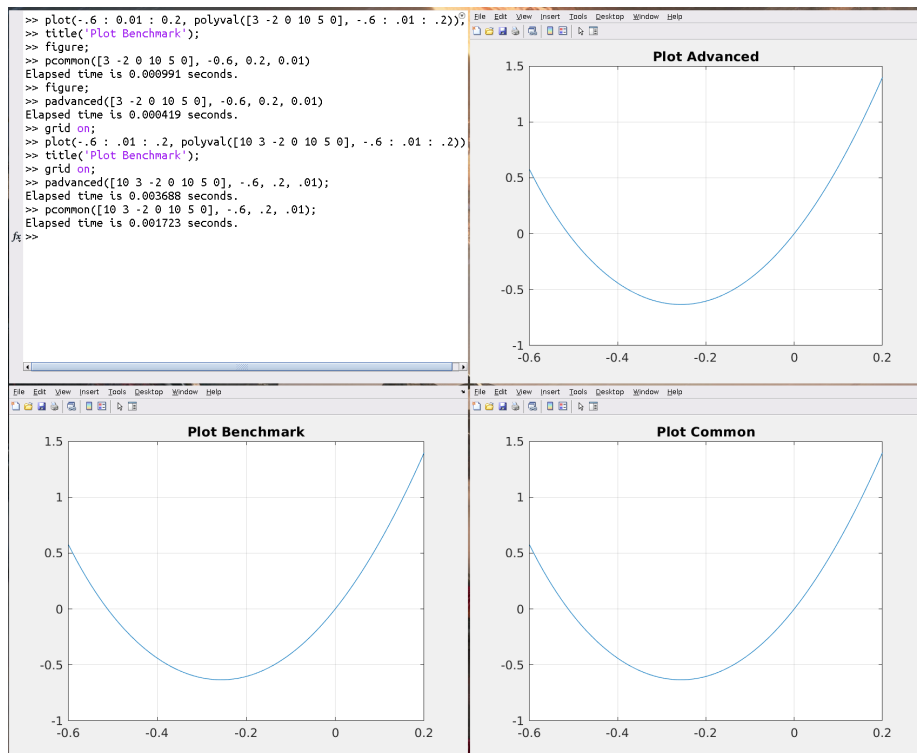
Listing 2: Polynom Advanced

```
1 function padvanced(A, a, b, h)
2
3     if nargin == 3
4         h = 0.01;
5     end
6
7     tic
8
9     % to the power vector
10    p1 = size(A,2) - 1 : -1 : 0;
11
12    % needed to transform Y
13    p2 = ones(size(A))';
14
15    % x values
16    Y1 = (a : h : b) .* p2;
17
18    % power every value
19    Y2 = Y1' .^ p1;
20
21    % multiply every value with input array
22    Y3 = Y2 .* A;
23
24    % calculate final p(x)
25    Y4 = sum(Y3, 2);
26
27    toc
28
29    plot(a : h : b, Y4);
30
31    title('Plot Advanced');
32
33    grid on;
34
35 end
```

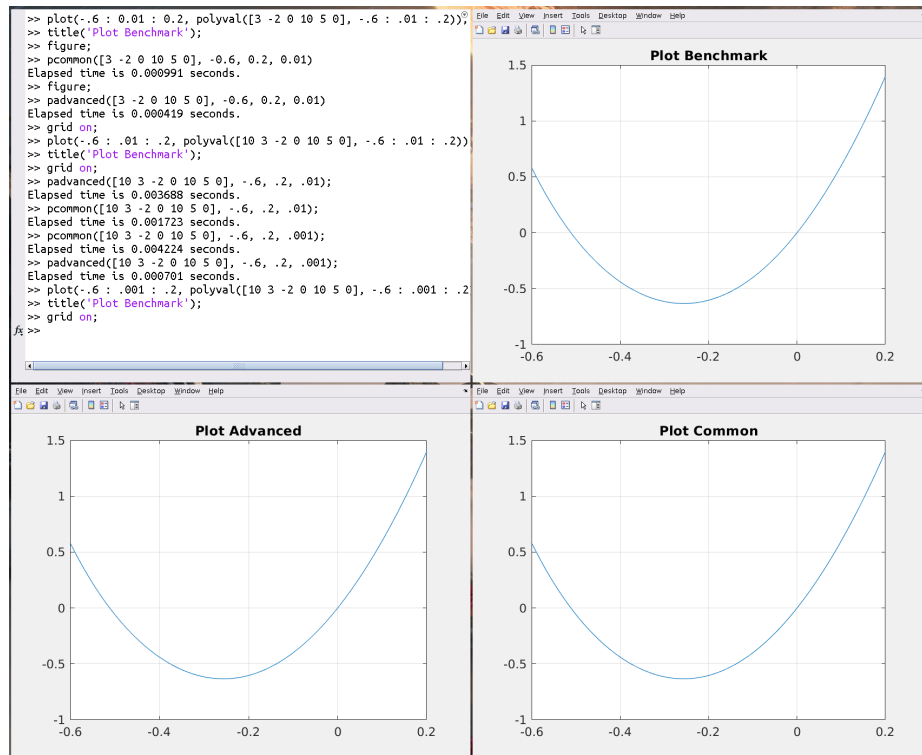
3. Run 1



4. Run 2



5. Run 3



6. Comparison:

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

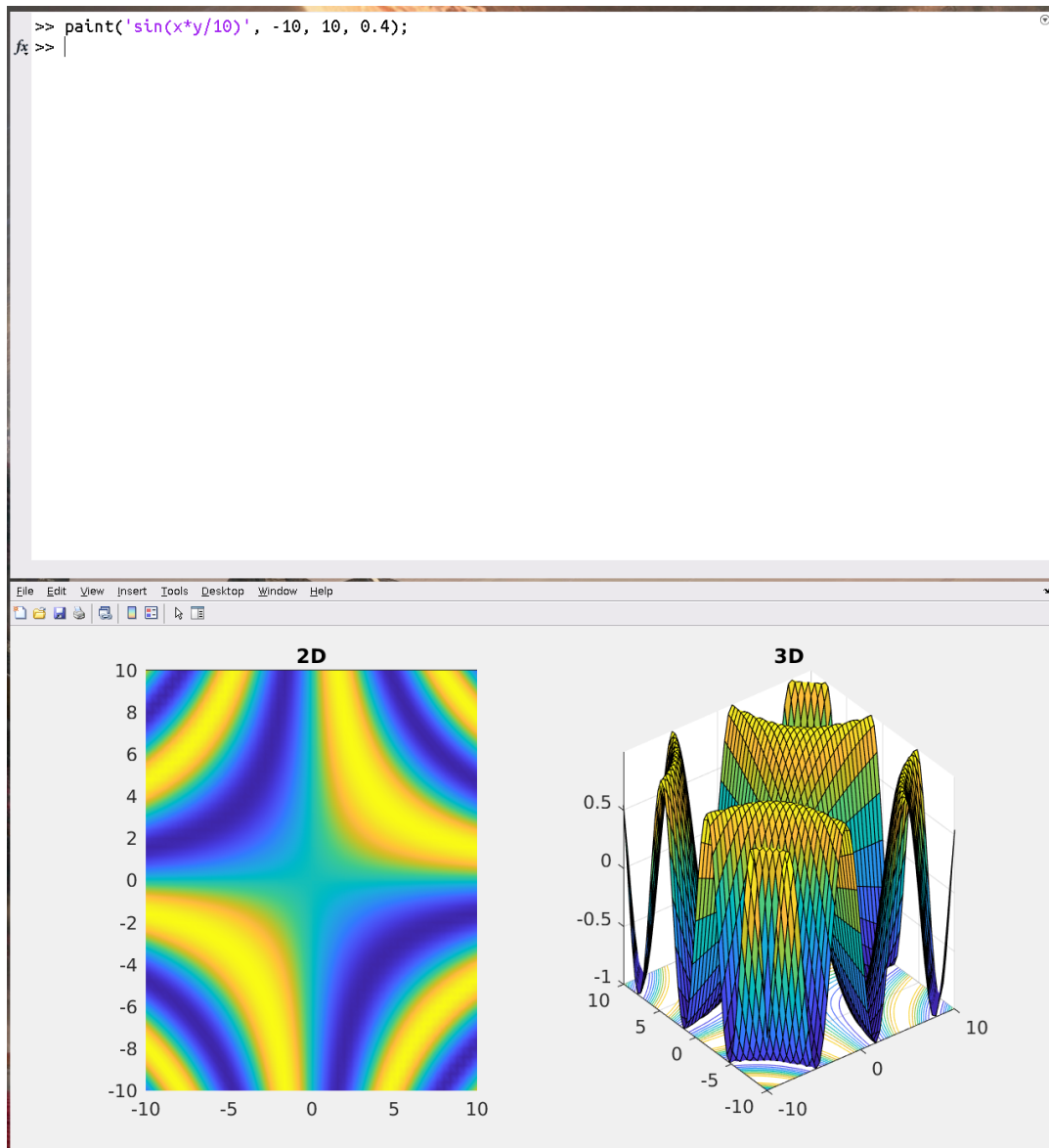
Aufgabe 2: Malen nach Zahlen (Pt 1)

1. Source Code:

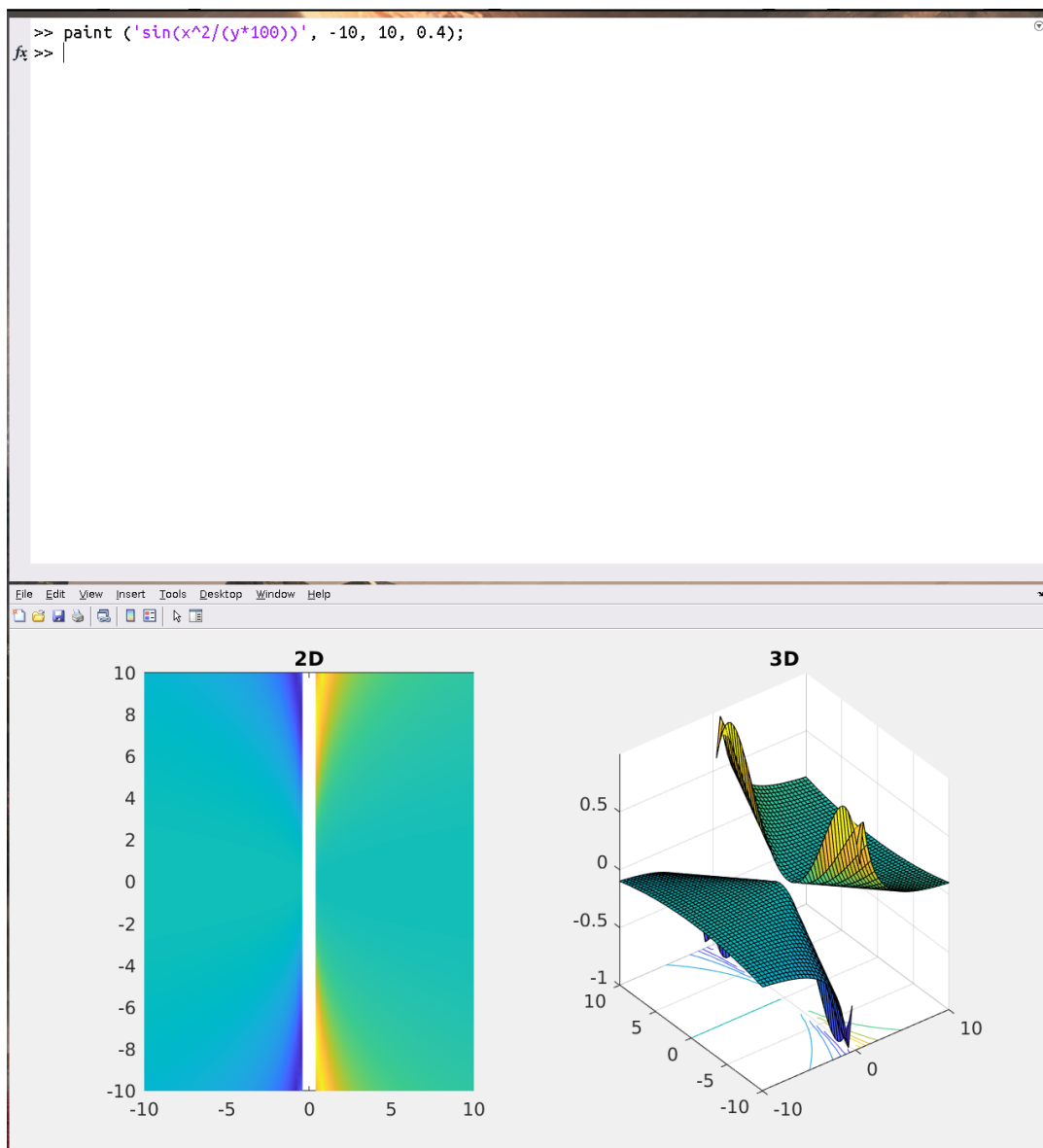
Listing 3: paint.m

```
1 function paint(fnstr, a, b, h)
2
3     if nargin == 3
4         h = 0.2;
5     end
6
7     N = (b - a) / h + 1;
8
9     Z = zeros(N);
10
11     x_ = 0;
12
13     for x = a : h : b
14         x_ = x_ + 1;
15         y_ = 0;
16         for y = a : h : b
17             y_ = y_ + 1;
18             Z(x_, y_) = eval(fnstr);
19         end
20     end
21
22     tiledlayout(1, 2);
23     nexttile;
24     pcolor(a : h : b, a : h : b, Z);
25     shading interp;
26     title(2D);
27     nexttile;
28     surf(a : h : b, a : h : b, Z);
29     title(3D);
30
31
32 end
```

2. Run 1



3. Run 2



Aufgabe 3: Malen nach Zahlen (Pt 2)

1. Source Code:

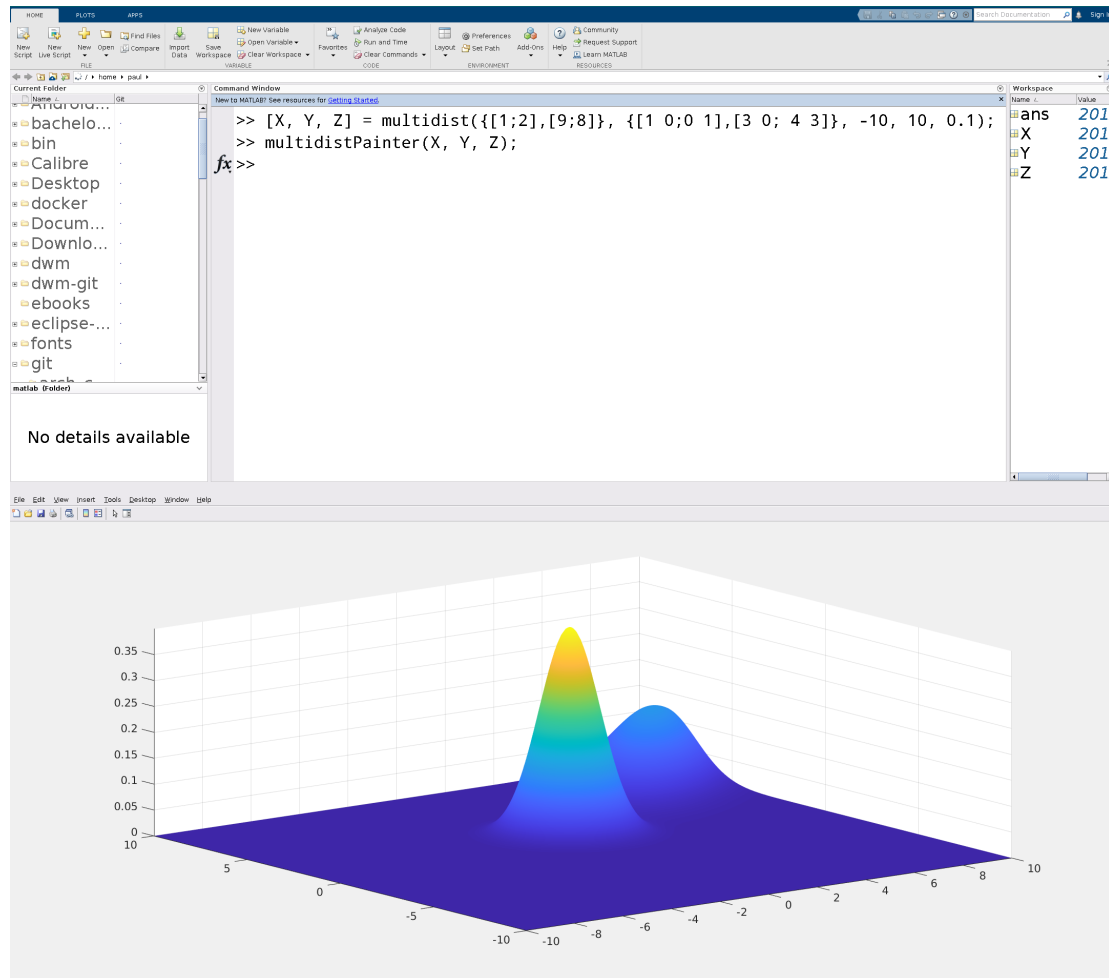
Listing 4: multidist.m

```
1 function [Xout, Yout, Zout] = multidist(mu, sigma, a, b, h)
2
3     if nargin == 4
4         h = 0.01;
5     end
6
7     N = (b - a) / h + 1;
8     x = zeros(N);
9     y = zeros(N);
10    z = zeros(N);
11    xc = 0;
12
13    for x_ = a : h : b
14        yc = 0;
15        xc = xc + 1;
16        for y_ = a : h : b
17            yc = yc + 1;
18            x(xc, yc) = x_;
19            y(xc, yc) = y_;
20            X = [x_; y_];
21            z(xc, yc) = 0;
22            for i = 1 : 1 : size(mu, 2)
23                p1 = 1 / ((2 * pi)^(size(mu{i}, 2) / 2) * det(sigma{i})
24                    ^0.5);
25                p2 = exp(-0.5*(X - mu{i})' * inv(sigma{i}) * (X - mu{i}))
26                    ;
27                p_x = p1 * p2;
28                z(xc, yc) = z(xc, yc) + p_x;
29            end
30        end
31    end
32
33    Xout = x;
34    Yout = y;
35    Zout = z;
36
37    end
```

Listing 5: multidistPainter.m

```
1 function multidistPainter(X, Y, Z)
2     surf(X, Y, Z);
3     shading interp;
4 end
```


2. Run 1



3. Run 2

