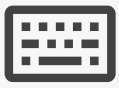
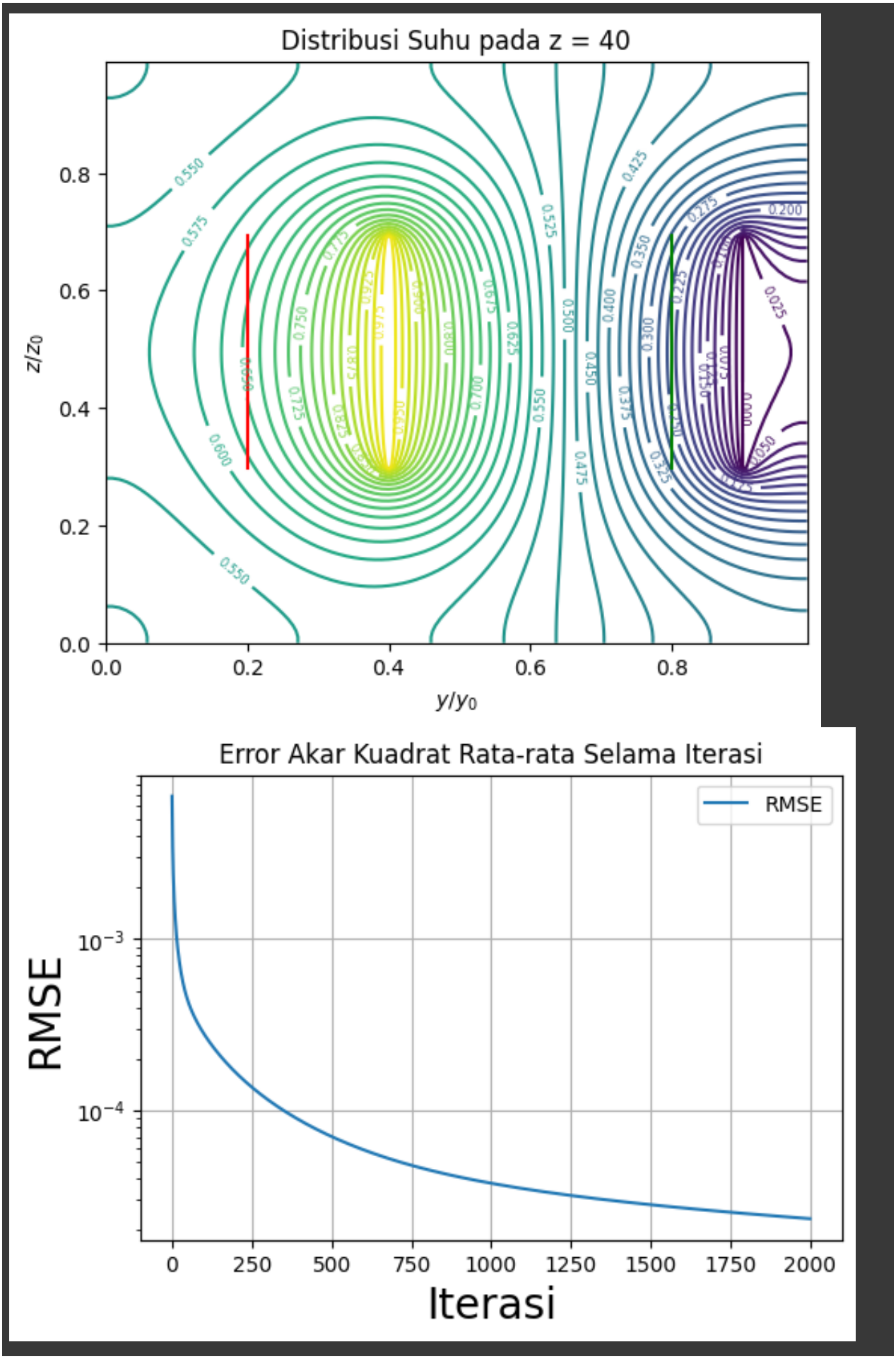
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve, generate_binary_structure

# Inisialisasi parameter
N = 100
grid = np.zeros((N, N, N)) + 0.5  # Suhu awal
grid[30:70, 30:70, 40] = 1  # Area panas di z = 40
grid[30:70, 30:70, 90] = 0  # Area dingin di z = 90
mask_pos = grid == 1  # Mask untuk area panas
mask_neg = grid == 0  # Mask untuk area dingin

# Buat kernel konvolusi
kern = generate_binary_structure(3, 1).astype(float) / 6
kern[1, 1, 1] = 0  # Nilai pusat adalah nol

# Fungsi untuk kondisi batas Neumann
def neumann(a):
    a[0, :, :] = a[1, :, :]
    a[-1, :, :] = a[-2, r:, :]
    a[:, 0, :] = a[:, 1, :]
    a[:, -1, :] = a[:, -2, :]
    a[:, :, 0] = a[:, :, 1]
    a[:, :, -1] = a[:, :, -2]
    return a

# Pengaturan iterasi
err = []
iters = 2000
for i in range(iters):
    grid_updated = convolve(grid, kern, mode='constant')
    # Terapkan kondisi batas Neumann
    grid_updated = neumann(grid_updated)
    # Terapkan kondisi batas Dirichlet
    grid_updated[mask_pos] = 1
    grid_updated[mask_neg] = 0
    # Hitung error
    err.append(np.mean((grid - grid_updated) ** 2))
    grid = grid_updated

# Visualisasi distribusi suhu
slc = 40
plt.figure(figsize=(6, 5))
CS = plt.contour(np.arange(100) / 100, np.arange(100) / 100, grid[slc], le
plt.clabel(CS, CS.levels, inline=True, fontsize=6)
plt.xlabel('$y/y_0$')
plt.ylabel('$z/z_0$')
```

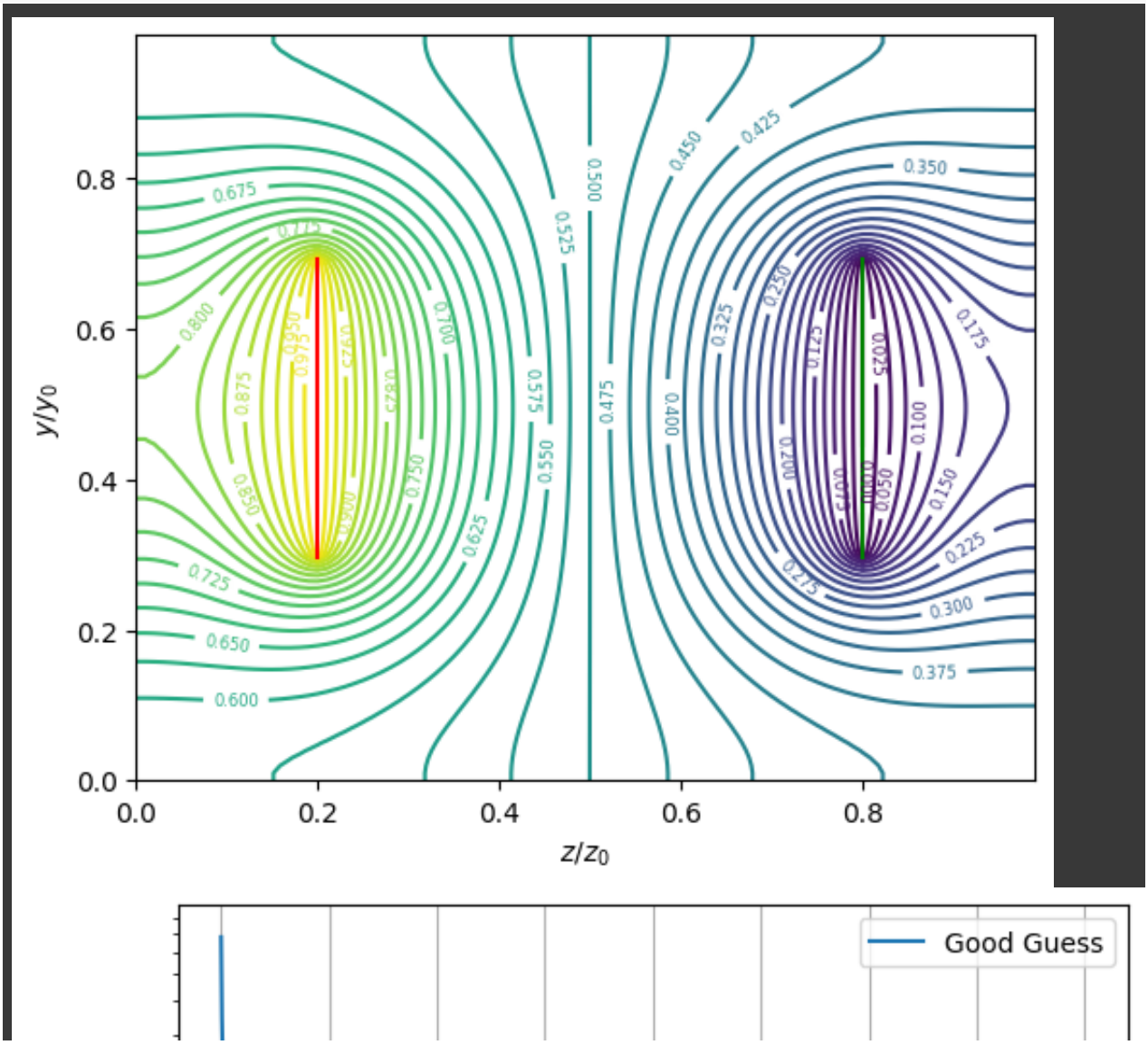
```
plt.ylabel('$z/z_0$')
plt.axvline(0.2, ymin=0.3, ymax=0.7, color='r')
plt.axvline(0.8, ymin=0.3, ymax=0.7, color='g')
plt.title('Distribusi Suhu pada z = 40')
plt.show()

# Plot RMSE selama iterasi
plt.figure(figsize=(6, 4))
plt.semilogy(np.sqrt(np.array(err)), label='RMSE')
plt.legend()
plt.xlabel('Iterasi', fontsize=20)
plt.ylabel('RMSE', fontsize=20)
plt.title('Error Akar Kuadrat Rata-rata Selama Iterasi')
plt.grid()
plt.show()
```

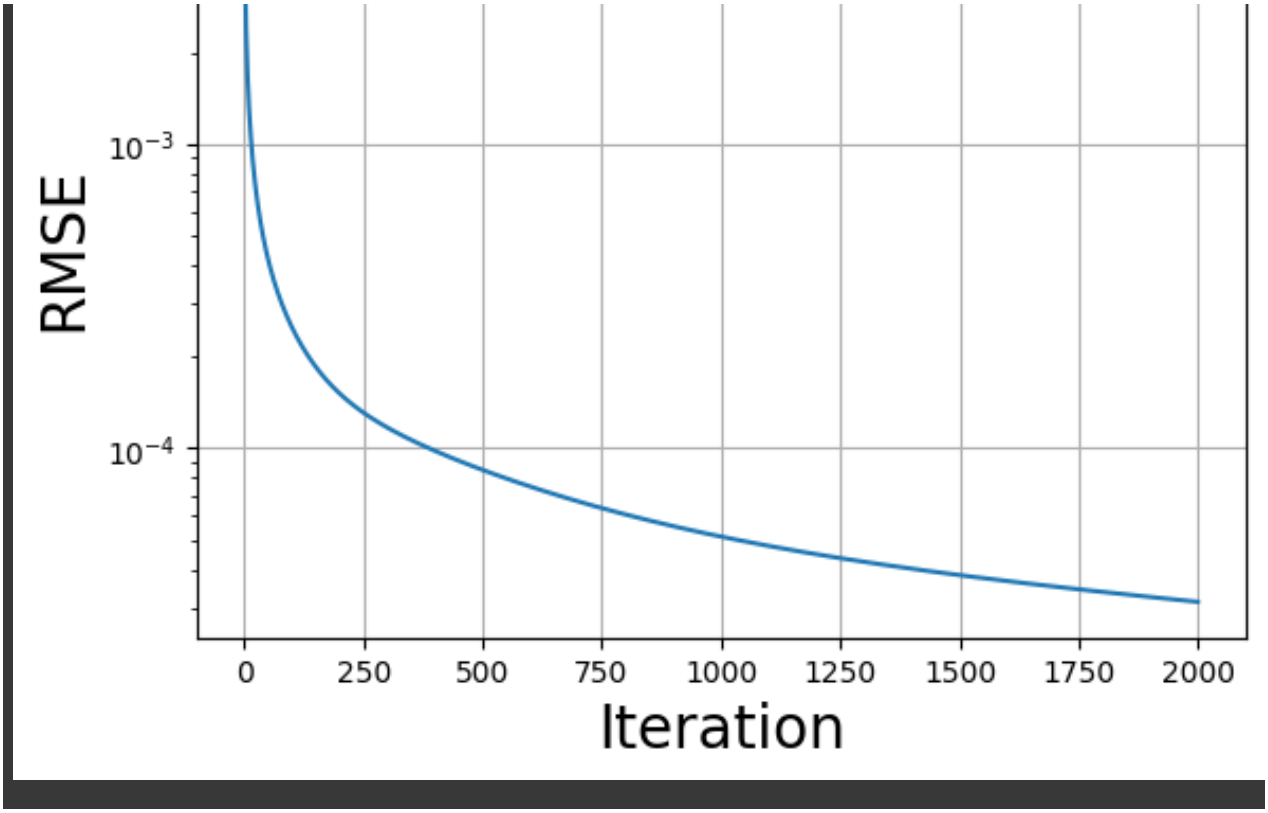



Distribusi Suhu pada z = 40
z/z_0
y/y_0
0.0
0.2
0.4
0.6
0.8
Error Akar Kuadrat Rata-rata Selama Iterasi
RMSE
Iterasi
10^{-3}
10^{-4}
0
250
500
750
1000
1250
1500
1750
2000

```
import numpy as np
import matplotlib.pyplot as plt
# plt.style.use(['science','notebook'])
from scipy.ndimage import convolve, generate_binary_structure

# Inisialisasi parameter
N = 100
grid = np.zeros((N, N, N)) + 0.5
grid[30:70, 30:70, 20] = 1
grid[30:70, 30:70, 80] = 0

# Mask untuk area panas dan dingin
mask_pos = grid == 1
mask_neg = grid == 0

# Buat meshgrid
yv, xv, zv = np.meshgrid(np.arange(N), np.arange(N), np.arange(N))

# Buat kernel konvolusi
kern = generate_binary_structure(3, 1).astype(float) / 6
kern[1, 1, 1] = 0

# Fungsi untuk kondisi batas Neumann
def neumann(a):
    a[0, :, :] = a[1, :, :]
    a[-1, :, :] = a[-2, :, :]
    a[:, 0, :] = a[:, 1, :]
    a[:, -1, :] = a[:, -2, :]
    a[:, :, 0] = a[:, :, 1]
    a[:, :, -1] = a[:, :, -2]
    return a

# Iterasi untuk pembaruan grid
err = []
iters = 2000
for i in range(iters):
    grid_updated = convolve(grid, kern, mode='constant')

    # Boundary conditions (Neumann)
    grid_updated = neumann(grid_updated)

    # Boundary conditions (Dirichlet)
    grid_updated[mask_pos] = 1
    grid_updated[mask_neg] = 0

    # Hitung error antara grid lama dan baru
    err.append(np.mean((grid - grid_updated) ** 2))
```

```
    grid = grid_updated

# Visualisasi hasil
slc = 40
plt.figure(figsize=(6, 5))
CS = plt.contour(np.arange(100) / 100, np.arange(100) / 100, grid[slc], le
plt.clabel(CS, CS.levels, inline=True, fontsize=6)
plt.xlabel('$z/z_0$')
plt.ylabel('$y/y_0$')
plt.axvline(0.2, ymin=0.3, ymax=0.7, color='r')
plt.axvline(0.8, ymin=0.3, ymax=0.7, color='g')
plt.show()

# Plot RMSE
plt.semilogy(np.sqrt(np.array(err)), label='Good Guess')
plt.legend()
plt.xlabel('Iteration', fontsize=20)
plt.ylabel(r'RMSE', fontsize=20)
plt.grid()
```




RMSE
10^{-3}
10^{-4}
0
250
500
750
1000
1250
1500
1750
2000
Iteration