Listing 1: Neural network

```
# neural_network.m
# Neural network implementation for searching convex function minimum with convex
    constraints.
# Vladimir Rutsky <altsysrq@gmail.com>
# 26.05.2009

f  = inline("x^2 + y^2 - 10 * x - 8 * y");
g1 = inline("    x + 2 * y - 2"); # <= 0
g2 = inline("2 * x +     y - 2"); # <= 0

precision = 1e-4;
startX = 20;
startY = 20;
startStep = 5;
neuronsHalfX = 10;
neuronsHalfY = 10;

function eax = processNetworkLayer( f, g1, g2, centerX, centerY, stepX, stepY,
    neuronsHalfX, neuronsHalfY )
  assert(stepX > 0);
  assert(stepY > 0);

  assert(neuronsHalfX > 0);
  assert(neuronsHalfY > 0);

  x0 = centerX - neuronsHalfX * stepX;
  x1 = centerX + neuronsHalfX * stepX;
  y0 = centerY - neuronsHalfY * stepY;
  y1 = centerY + neuronsHalfY * stepY;

  minFound = 0;
  minX = x0;
  minY = y0;
  minFuncVal = f(minX, minY);


  # Iterating through all neurons
  for x = x0:stepX:x1
    for y = y0:stepY:y1
      if (g1(x, y) <= 0 && g2(x, y) <= 0)
        # Feasible point <=> weight coefficient is not infinity.
        funcVal = f(x, y);
        if (!minFound || funcVal < minFuncVal)
          minFound = 1;

          # More chances that current neuron will be activated.
          minX = x;
          minY = y;
          minFuncVal = funcVal;
        endif
      endif
    endfor
  endfor

  eax =  [minX, minY];
endfunction

step = startStep;
centerX = startX;
centerY = startY;

nSteps = 0;
while (step >= precision)
  newCenter = processNetworkLayer(f, g1, g2, centerX, centerY, step, step, neuronsHalfX,
      neuronsHalfY);
  centerX = newCenter(1);
  centerY = newCenter(2);
  step = step ./ 2;
  nSteps = nSteps + 1;
endwhile
```

```
68
69 printf("Found minimum at (%g, %g) with precision %.1e after %d steps.", centerX, centerY,
       precision, nSteps);
```