Supplementary Information for Fast and Simple Super-resolution with Single Images

Paul H.C. Eilers Cyril Ruckebusch

May 23, 2022

Here we present details of the conjugate gradients (CG) algorithm and its Matlab implementation, based on the Wikipedia lemma (`https://en.wikipedia.org/wiki/Conjugate_gradient_method`). We start with the one-dimensional case. The PSF is represented by the m-by-n matrix S and the observed "image" is the m-vector$y$. We have to solve the system of penalized least squares equations $(S'S + \kappa I + \lambda D'D)x = S'y$ for the n-vector x. For efficiency, it is re-written as $Gx = u$.

Listing 1: Code for standard conjugate gradients algorithm.

```
% Set up ridge regression
n = size(S, 2);
kappa = 0.001;
D = diff(eye, n);
lambda = 1;
G = S' * S + kappa * eye(n) + lambda * D' * D;
u = S' * y;

% Initialize for conjugate gradients
x = zeros(n, 1);
r = u - G * x;
p = r;

% Iterate CG
for it = 1:50

% Update p and r
```

```
18    q = G * p;
19    alpha = (r' * r) / (p' * q);
20    x = x + alpha * p;
21    rnew = r - alpha * q;
22    beta = (rnew' * rnew) / (r' * r);
23    r = rnew;
24    p = r + beta * p;
25
26    % Monitor convergence
27    err = sqrt(mean((u - G * x) .^ 2 ));
28    disp([it log10(err)])
29    if err < 1e-3
30    break
31    end
32    end
```

The core of the algorithm takes only seven lines. One operation is a matrix-vector product (`q = G * p`), all others work on only vectors.

Listing 2: Code for two-dimensional conjugate gradients algorithm.

```
1     % Prepare components of linear system
2     U = S1' * Y * S2;
3     G1 = S1' * S1;
4     G2 = S2' * S2;
5     kappa = 1;
6     lambda = 1;
7     D1 = diff(eye(n1));
8     D2 = diff(eye(n2));
9     V1 = lambda * D1' * D1;
10    V2 = lambda * D2' * D2;
11
12    % Initialize for conjugate gradients
13    R = U;
14    P = R;
15    n1 = size(G1, 2);
16    n2 = size(G2, 2);
17    X = zeros(n1, n2);
18
19    for it = 1:100
20
```

```
    % Update P and R
    Q = G1 * P * G2 + kappa * P + V1 * P + P * V2';
    alpha = sum(R(:) .^ 2) / sum(P(:) .* Q(:));
    X = X + alpha * P;
    Rnew = R - alpha * Q;
    rs1 = sum(R(:) .^ 2);
    rs2 = sum(Rnew(:) .^ 2);
    beta = rs2 / rs1;
    P = Rnew + beta * P;
    R = Rnew;

    % Monitor convergence
    rms = sqrt(rs1 / (n1 * n2));
    disp([it log10(rms)])
    if rms < 1e-3
    break
    end
    end
```