REU Week 2 Writeup

Sayem Hoque

June 26, 2017

1 Perona Malik

Perona Malik is an anisotropic diffusion that can be applied to images. The goal behind perona malik is to reduce the noise in an image without ruining essential information from the image, like edges. A key feature oredf the perona malik algorithm is a monotonically decreasing "g" function. There are a few different g functions that have been implemented in the perona malik code,

```
'peronamalik.m'

function u = peronamalik(u, iterations)



a = 10;
u = double(u);

dt = 0.1;
h = 1;
[r,c] = size(u);

for t = 1:iterations

        uxf = (u(:,[2:c,c]) - u) ./ h;
        uyf = (u([2:r,r],:) - u) ./ h;

        %uxb = (u - u(:, [1, 1:c-1])) ./ h;
        %uyb = (u - u([1, 1:r-1], :)) ./ h;

        p = ((uxf.^2 + uyf.^2).^0.5);
        % p = ((uxb.^2 + uyb.^2).^0.5);
        gu = 1 ./ (1 + p.^2 ./ a.^2);

        %gu = 1 ./ ((1 + p.^2) ^0.5);

        guxf = gu .* uxf;
        guyf = gu .* uyf;
```

```
        guxfb = (guxf - guxf(:, [1, 1:c-1])) ./ h;
        guyfb = (guyf - guyf([1,1:r-1], :)) ./ h;

        u_out = u + dt .* (guxfb + guyfb);
        u = u_out;
end

end
```

There are a few g functions 'gu'. $1/(1+p^2)$ and $1/(1+p^2/(a^2))$ were implemented in the code. The one with the parameter a is currently being used because it allowed for more flexibility in the decreasing behavior of the function.

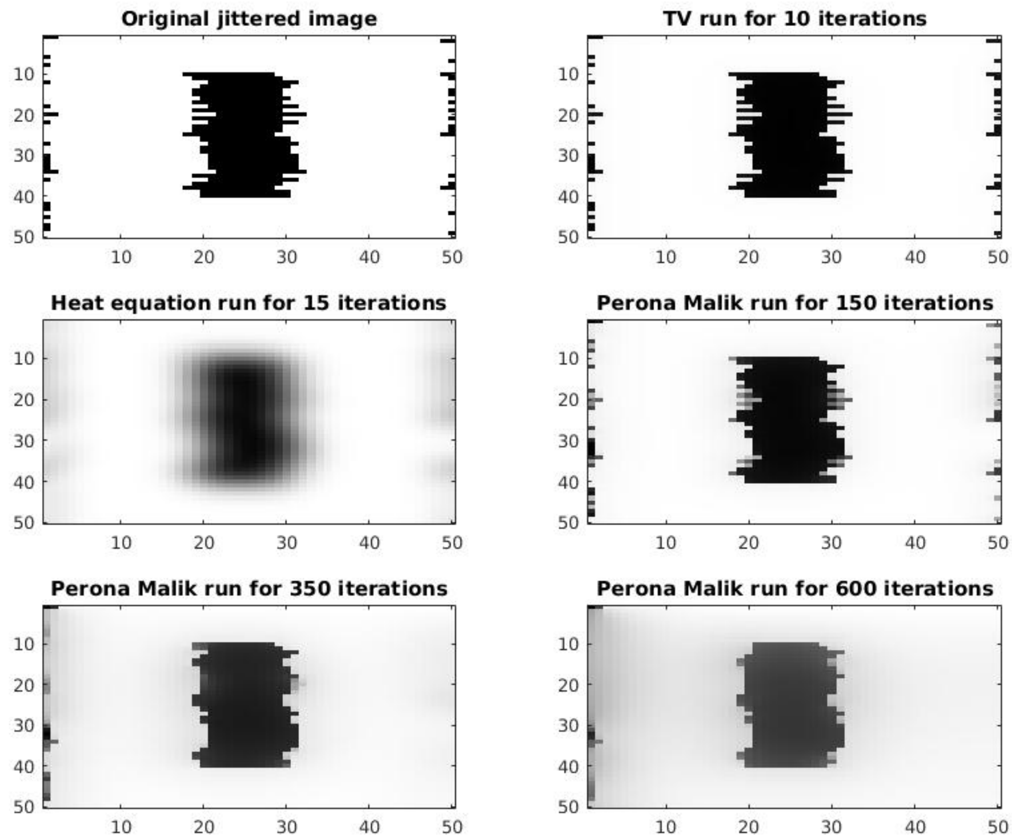
After each image is given a random jitter, Heat Equation, TVR, and perona malik at different iterations are compared. The hope is that even with the jittering of the image, the major features in the image will allow the anisotropic diffusion to let those features become more apparent again.

1.1 Images with perona malik run

2 Newton's method

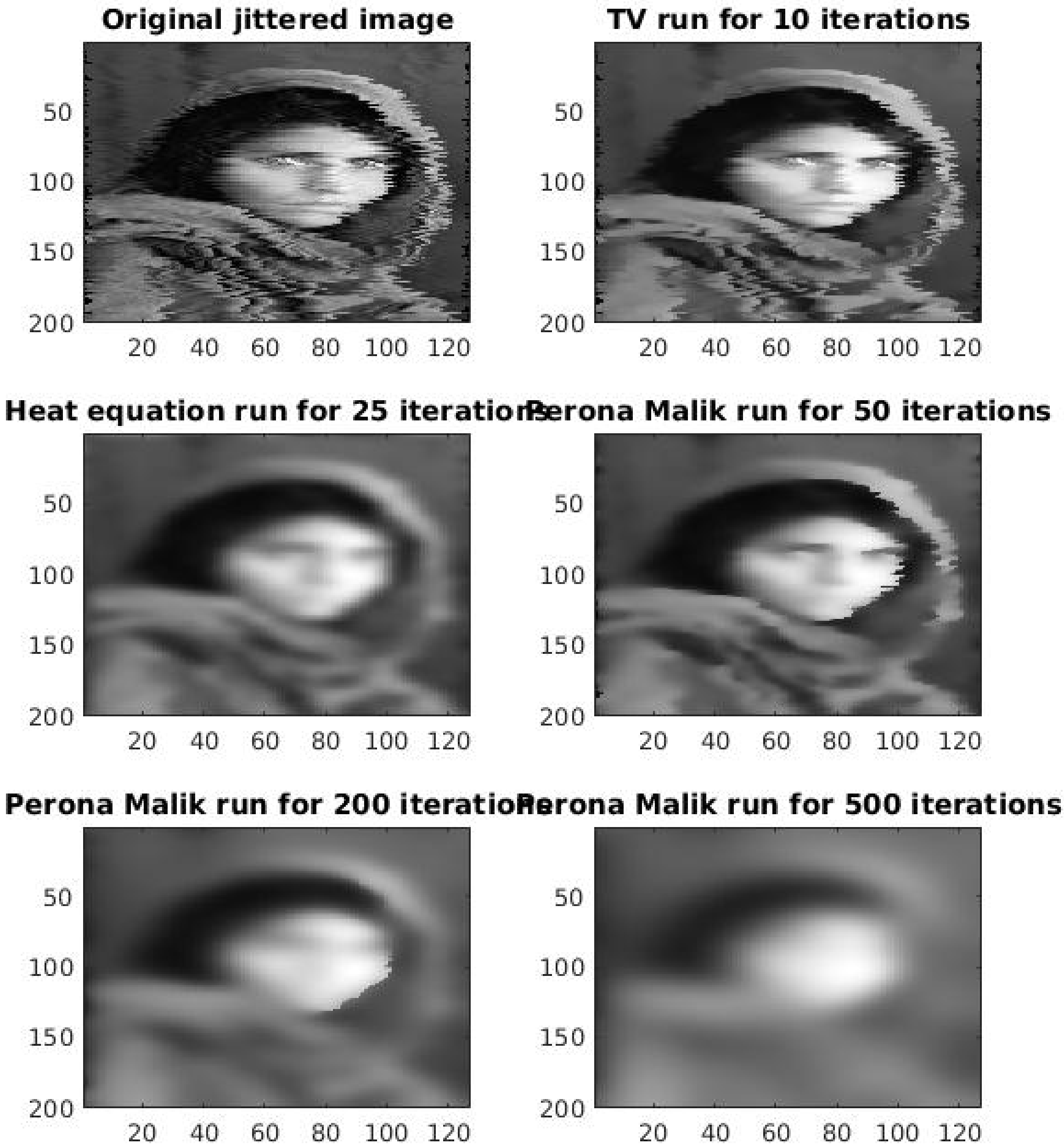
Steps: make an initial approximation of x close to c. Determine a new approximation using formula where x(n+1) = xn + p(xn)/p'(xn). Let this run until xn+1 - xn is less than a certain input accuracy.

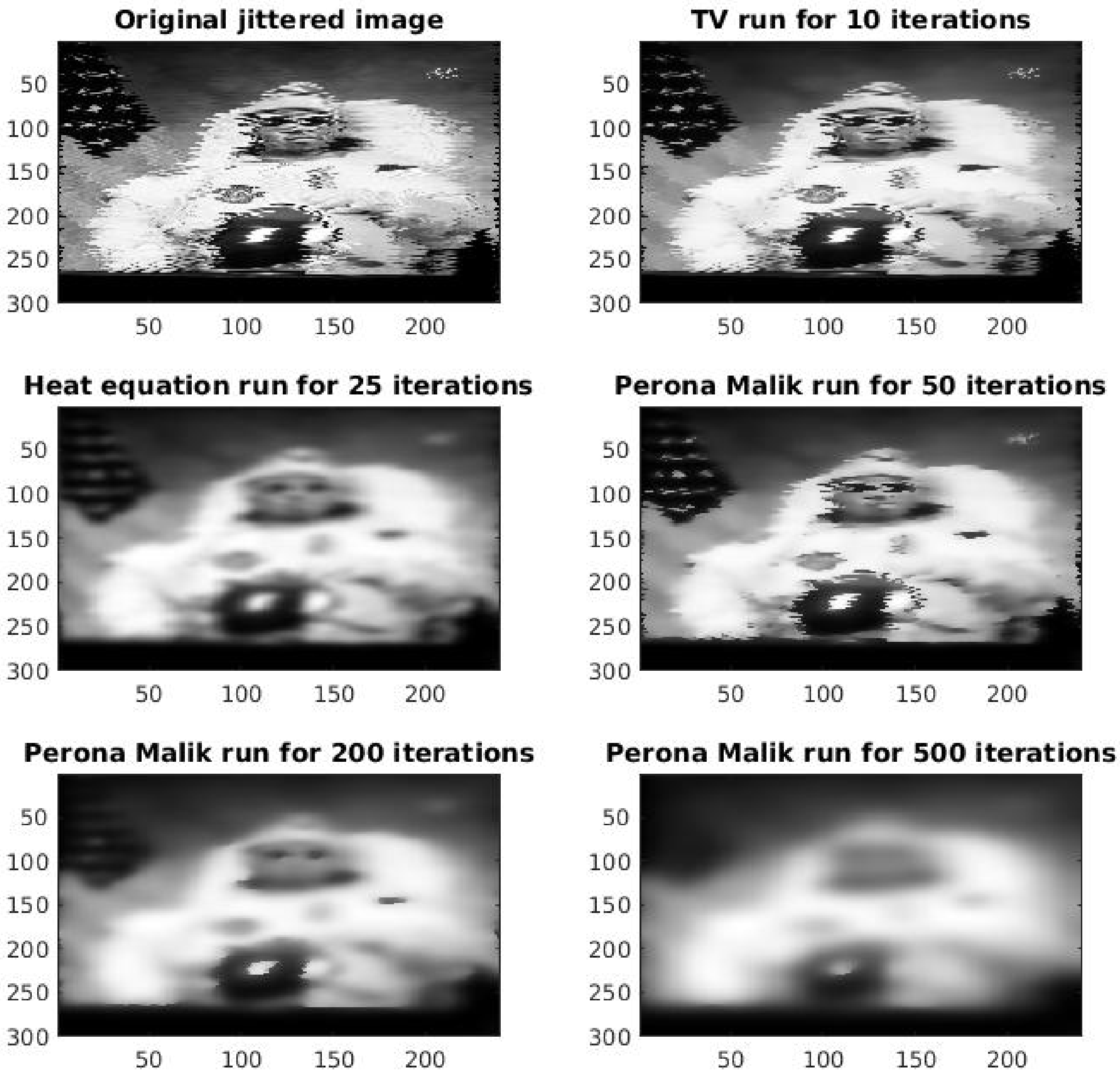
```
function [] = newton()
% calculate the roots of a polynomial using Newton's method.

% coefficient matrix
a = [2, 2]

% absolute error for Newton iteration
abserr = 0.5

% maximum iterations
itmax = 10

% initial condition
init = 1

x = linspace(-100,100,200);
f = polyval(a, x);

plot(x, f);

%now use Newton's method to find a root
```




```
bool = 1;
xk = init;
xk1 = 0;
k = 1;

while(bool == 1)

   if(bool == 1)
        [p, pprime] = polyderiv(a);
        p =
        pprime =
        xvals(k) = xk;
```

```
        pvals(k) = p;
        kvals(k) = k;
        xk = xk - p / pprime;
        k = k+1;

   end

    if(k < itmax) && (abs(xk - xk1) > abserr)
        bool = 1;
    else
        bool = 0;
    end

end

iteration = (kvals)'
xvalue = (xvals)'
fvalue = (pvals)'
```


Original jittered image
TV run for 10 iterations
Heat equation run for 25 iterations
Perona Malik run for 50 iterations
Perona Malik run for 200 iterations
Perona Malik run for 500 iterations
50
100
150
200
20 40 60 80 100 120


Original jittered image
TV run for 10 iterations
Heat equation run for 25 iterations
Perona Malik run for 50 iterations
Perona Malik run for 200 iterations
Perona Malik run for 500 iterations
50
100
150
200
250
300
50
100
150
200