Homework 8

Math 3607, Autumn 2021

Marco LoPiccolo

Table of Contents

Problem 1.

1. (Low-rank approximation using SVD; image compression) Load `hubble_gray.jpg`, which is a grayscale image taken by the Hubble Space Telescope, convert it to a matrix of floating point pixel intensities, and then display the image in MATLAB by

```
A = imread('hubble_gray.jpg');
imshow(A);
```

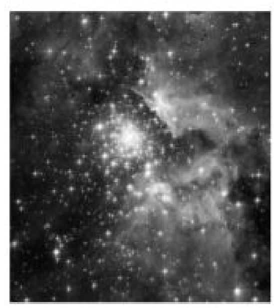


Figure 1: NGC 3603 (Hubble Space Telescope).

Following the demo in Lecture 23 as a guide,

(a) Plot the singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$ of A on a log scale (using `semilogy`).

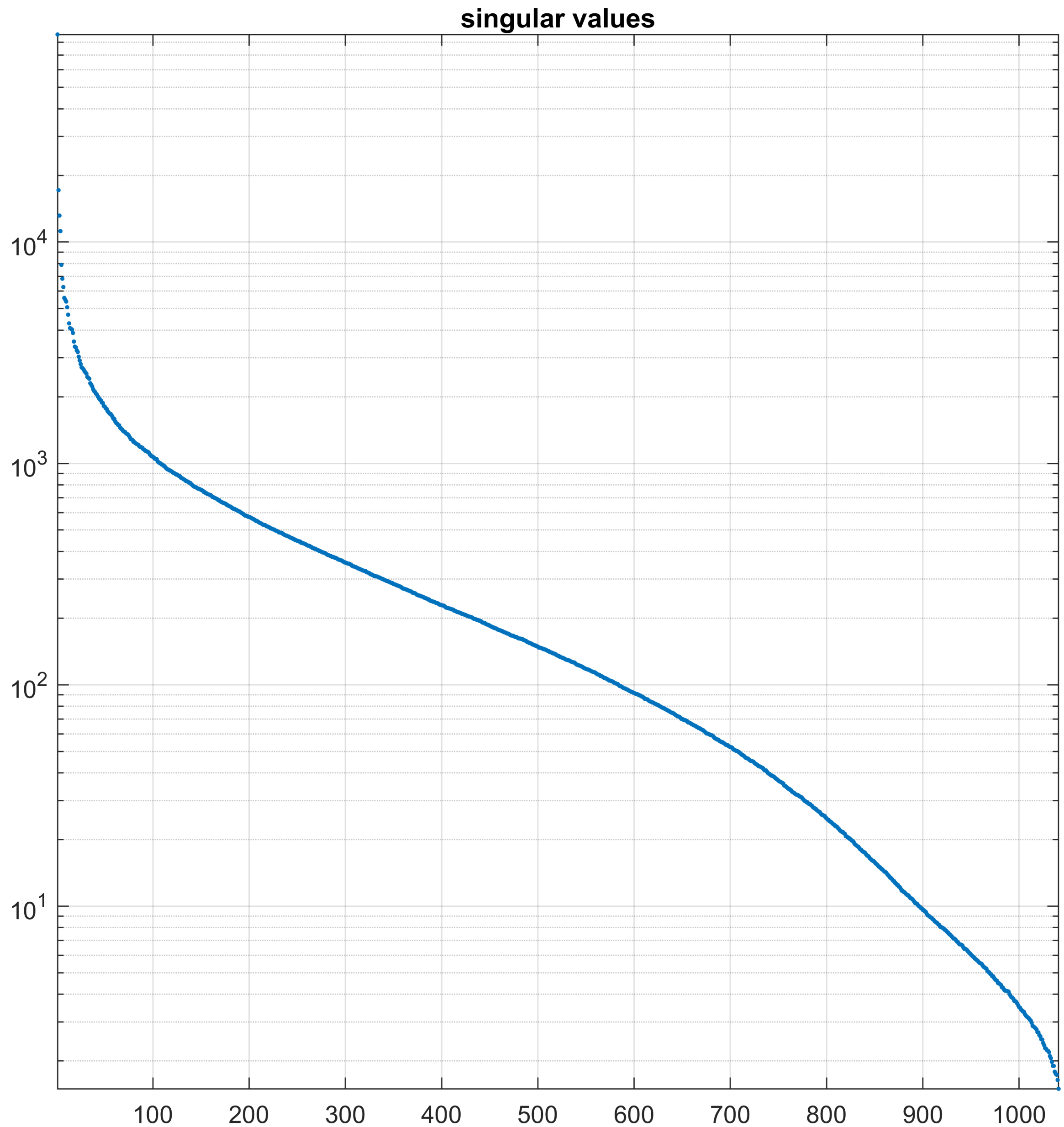
(b) Plot the accumulation of singular values of A.

(c) Compute the best approximations of A of rank $2, 20$, and 120 and display the corresponding images using `subplot`.

Part a.)

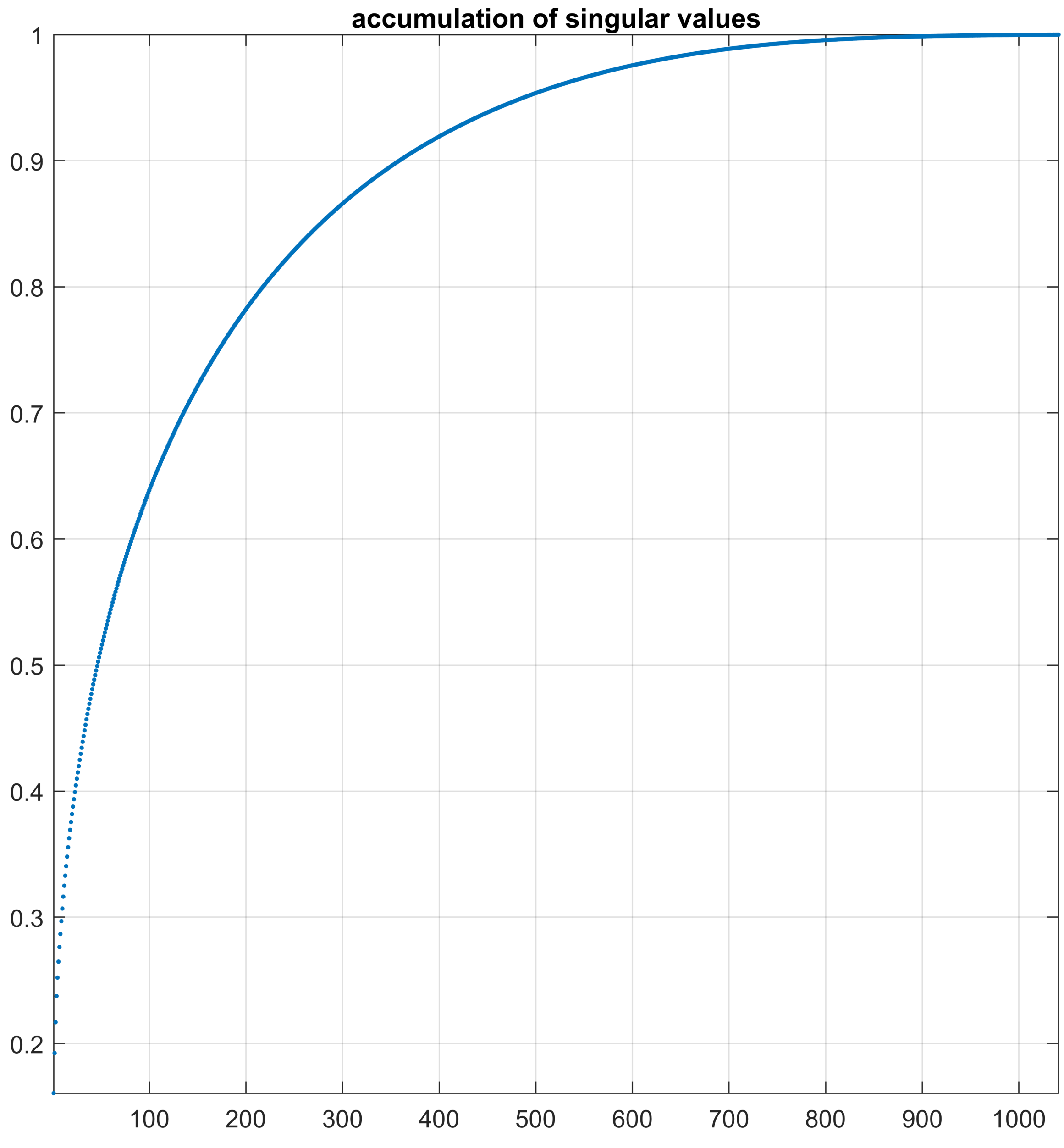
```
clf
load hubble_gray.jpg
A = imread('hubble_gray.jpg');
imshow(A);
```

```
A = double(A);
[U,S,V] = svd(A);
sigma = diag(S);
clf
semilogy(1:length(sigma), sigma, '.')
title('singular values'), axis tight, grid on
```



Part b.)

```
clf
plot(1:length(sigma), cumsum(sigma)/sum(sigma), '.')
title('accumulation of singular values'), axis tight, grid on
```

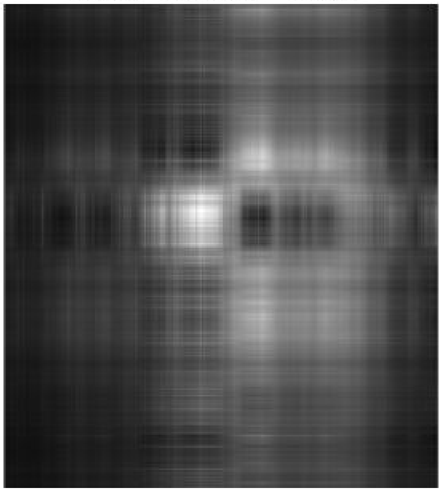


Part c.)

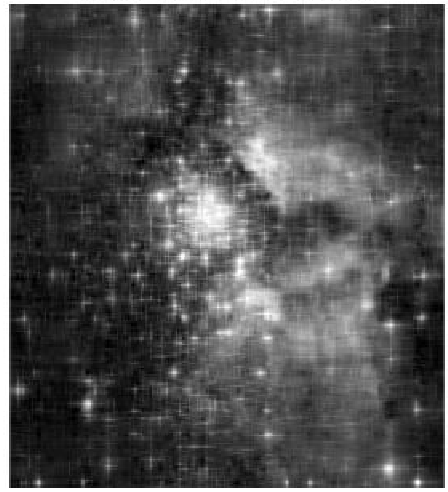
```
j = 0;
[m,n] = size(A);
clf
for k = [2 20 120]
    j = j + 1;
    subplot(2, 2, j)
    Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';  % rank-k approximation
    imshow(Ak, [0, 255])
    comp_ratio = k*(m+n+1)/(m*n);
    title(sprintf('rank = %d, ratio = %5.3f', k, comp_ratio))
```

```
end
```

rank = 2, ratio = 0.004



rank = 20, ratio = 0.037



rank = 120, ratio = 0.219

Problem 2.

2. (Annuity with `fzero`; **FNC** 4.1.4) A basic type of investment is an annuity: One makes monthly deposits of size P for n months at a fixed annual interest rate r, and at maturity collects the amount

$$\frac{12P}{r}\left(\left(1+\frac{r}{12}\right)^n - 1\right).$$

Say you want to create an annuity for a term of 300 months and final value of \$1,000,000. Using `fzero`, make a table of the interst rate you will need to get for each of the different contribution values $P = 500, 550, \ldots, 1000$.

```
n = 300;
finalValue = 1000000;
f = @(r, P) (12*P/r) * ((1+(r/12))^n - 1) - finalValue;
P = 500:50:1000;
r = zeros(size(P));
for j = 1:length(P)
    r(j) = fzero(@(r) f(r, P(j)), 1);
end
%use relative error for all the values in r
relerr = zeros(size(P));
for j = 1:length(P)
    relerr(j) = f(r(j),P(j)) / finalValue;
end
format long g
fprintf('                    P                Interest Rate                Relative Error')
```

```
                    P                Interest Rate                Relative Error
```

```
fprintf('-----------------------------------------------------------------------------')
```

```
-----------------------------------------------------------------------------
```

```
disp([P', r', relerr'])
```

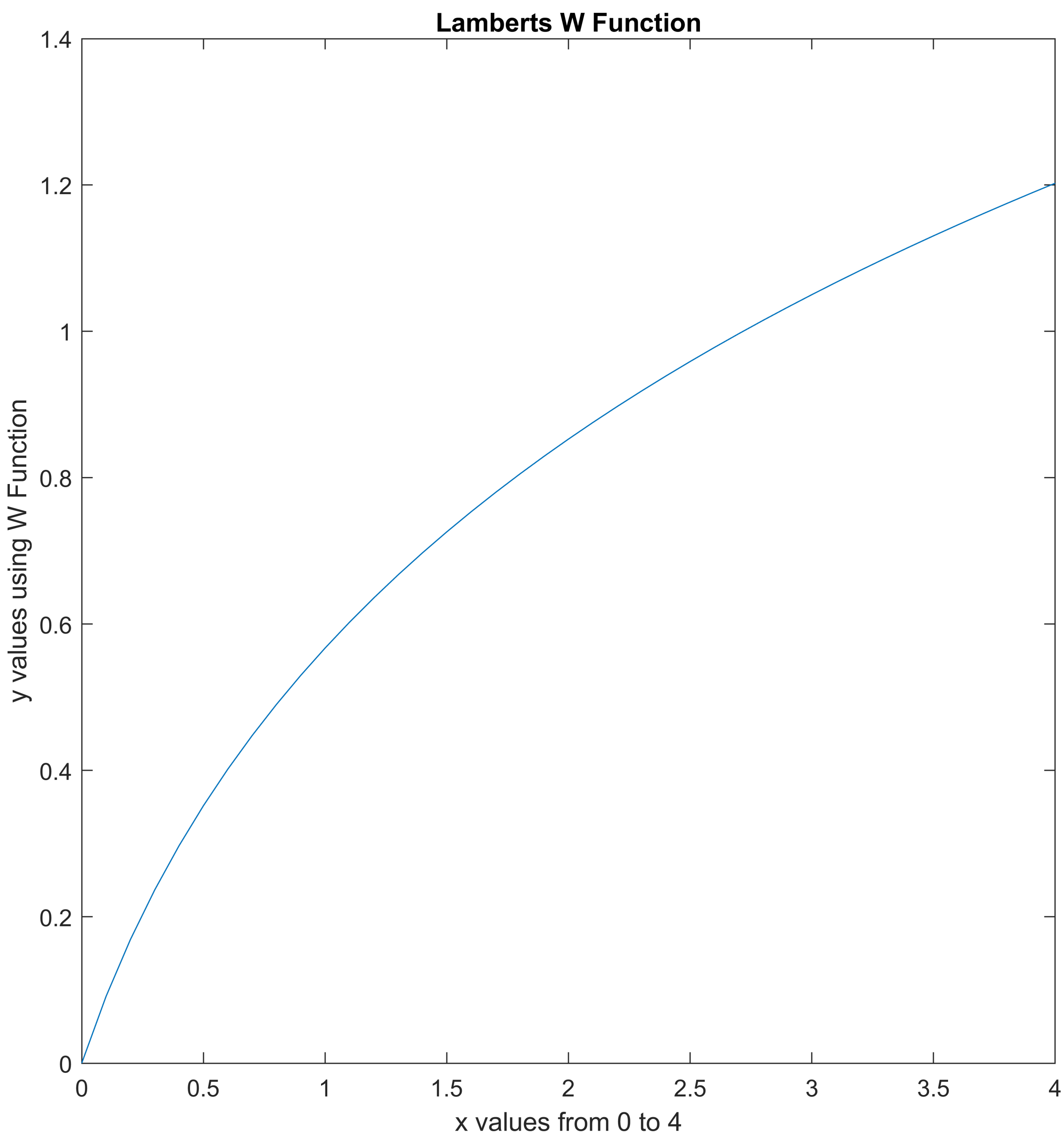
```
                       500         0.123511807047063      1.16415321826935e-14
                       550          0.11814632045792      1.93249434232712e-14
                       600         0.113200201561364     -2.15368345379829e-14
                       650          0.10860744853049     -2.94530764222145e-14
                       700         0.104316606191742      2.28174030780792e-14
                       750         0.100286770470683      1.50175765156746e-14
                       800        0.0964848788646416     -2.96859070658684e-14
                       850        0.0928838197313124      1.97906047105789e-14
                       900        0.0894610800540336     -9.77888703346252e-15
                       950        0.0861977573629547      3.00351530313492e-14
                      1000        0.0830778240105258      2.44472175836563e-14
```

Problem 3.

3. (Lambert's W function; **FNC** 4.1.6) Lambert's W function is defined as the inverse of xe^x. That is, $y = W(x)$ if and only if $x = ye^y$. Write a function `y = lambertW(x)` that computes W using `fzero`. Make a plot of $W(x)$ for $0 \le x \le 4$.

```
format long g
```

```
x = 0:0.1:4;
W = LambertW(x)';   %My function
K = lambertw(x)';   %Matlab's Built in Function to check
W - K;
clf
plot(x, W)
title('Lamberts W Function')
xlabel('x values from 0 to 4')
ylabel('y values using W Function')
```



Problem 4.

4. (Fixed-point iteration; adapted from **FNC** 4.2.1 and 4.2.2.) In each case below,

- $g(x) = \frac{1}{2}\left(x + \frac{9}{x}\right)$, $r = 3$.
- $g(x) = \pi + \frac{1}{4}\sin(x)$, $r = \pi$.
- $g(x) = x + 1 - \tan(x/4)$, $r = \pi$.

(a) Show that the given $g(x)$ has a fixed point at the given r and that fixed point iteration can converge to it.

(b) Apply fixed point iteration in MATLAB and use a log-linear graph (using `semilogy`) of the error to verify linear convergence. Then use numerical values of the error to determine an approximate value for the rate σ.

Part a.)

4a) $g(x) = \frac{1}{2}\left(x + \frac{9}{x}\right)$, $r = 3$

NTS: $3 = g(3)$

$$g(3) = \frac{1}{2}\left(3 + \frac{9}{3}\right) = \frac{1}{2}(3+3) = \frac{1}{2}\cdot 6 = 3$$

$$g(3) = 3 \checkmark$$

$$g'(x) = \frac{1}{2}\left(1 - 9x^{-2}\right)$$

$$g'(x) = \frac{1}{2} - \frac{9}{x^2}$$

$$g'(3) = \frac{1}{2} - \frac{9}{2(3^2)} = \frac{1}{2} - \frac{1}{2} = 0$$

$$|g'(3)| = 0 < 1$$

By theorem 3 in Lecture 25 the fixed point iterates generated by $x_{k+1} = g(x_k)$, $k = 1, 2, \dots,$ converge linearly with rate σ to the fixed point r for x_0 sufficiently close to r. Because the value is 0 it is superlinear.

$g(x) = \pi + \frac{1}{4}\sin(x)$, $r = \pi$

NTS: $g(\pi) = \pi$

$$g(\pi) = \pi + \frac{1}{4}\sin(\pi)$$

$$= \pi + \frac{1}{4}\cdot 0$$

$$= \pi = \pi \qquad g(\pi) = \pi \checkmark$$

$$g'(x) = \frac{1}{4}\cos(x)$$

$$g'(\pi) = \frac{1}{4}\cos(\pi) = \left| -\frac{1}{4} \right| = \frac{1}{4}$$

$$|g'(\pi)| = \frac{1}{4} < 1$$

By theorem 3 in Lecture 25 the fixed point iterates generated by $x_{k+1} = g(x_k)$, $k = 1, 2, \dots,$ converge linearly with rate σ to the fixed point r for x_0 sufficiently close to r.

$$g(x) = x + 1 - \tan(x/4), \quad r = \pi$$

NTS: $g(\pi) = \pi$

$$g(\pi) = \pi + 1 - \tan(\pi/4)$$
$$= \pi + 1 - 1 = \pi$$

Therefore $g(\pi) = \pi$ ✓

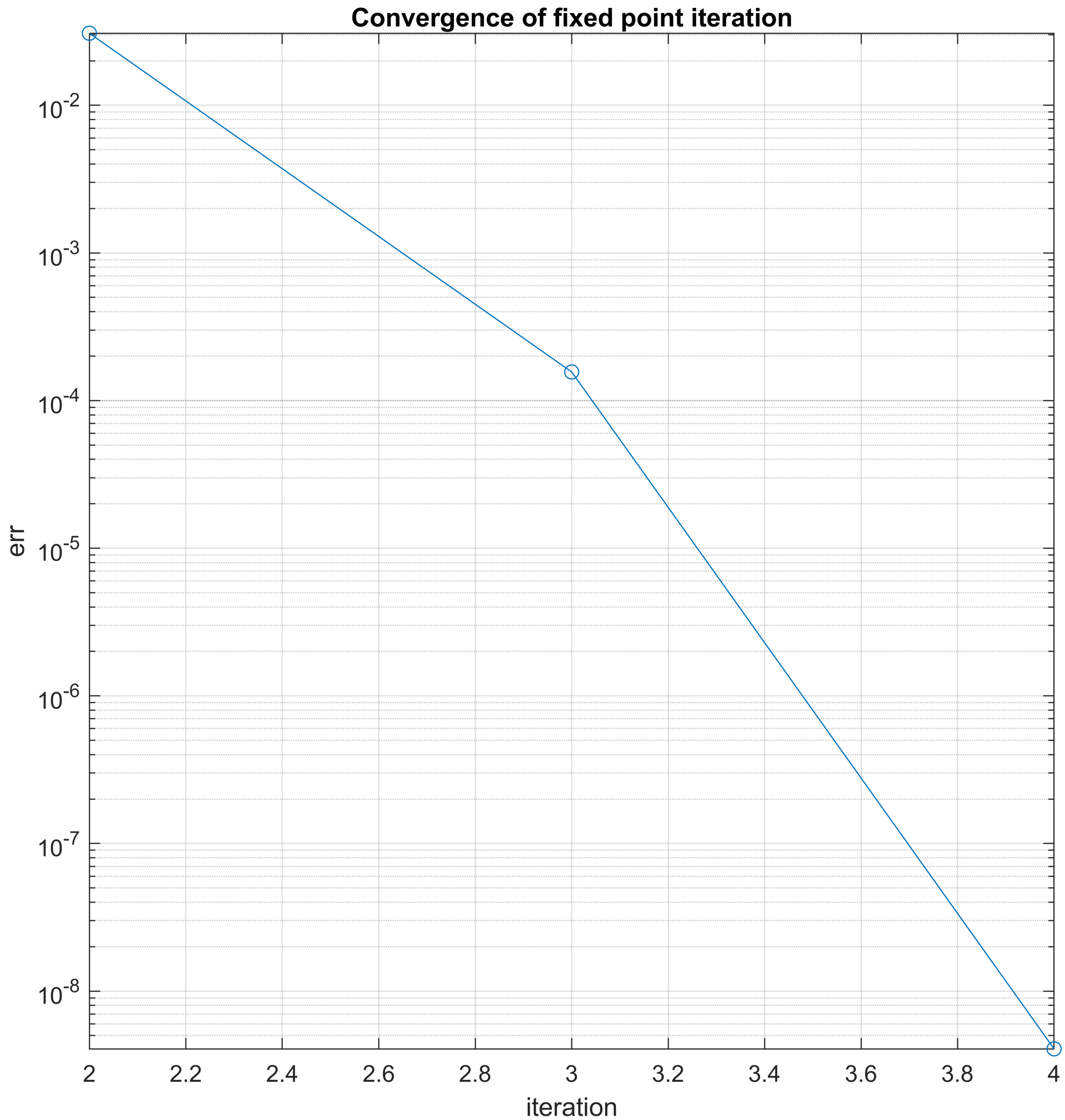
$$g'(x) = 1 - \frac{1}{4}\sec^2\left(\frac{x}{4}\right)$$
$$g'(\pi) = 1 - \frac{1}{4}\sec^2\left(\frac{\pi}{4}\right)$$
$$= 1 - \frac{1}{2} = \frac{1}{2} < 1$$
$$|g'(\pi)| = \frac{1}{2} < 1$$

By theorem 3 in Lecture 25 the fixed point iterates generated by $x_{k+1} = g(x_k)$, $k = 1, 2, \dots,$ converge linearly with rate σ to the fixed point r for x_0 sufficiently close to r.

Part b.)

```
g = @(x) (1/2) * (x+(9/x));
n = 10;
x = zeros(n, 1);
err = zeros(n,1);
```

```
r = 3;
x(1) = 2.6;
for k = 1:n-1
    x(k+1) = g(x(k));     %FPI
    err(k+1) = abs(x(k+1) - r);
end
clf
semilogy(1:10, err, 'o-')
xlabel('iteration')
ylabel('err')
title('Convergence of fixed point iteration')
axis tight, grid on
```

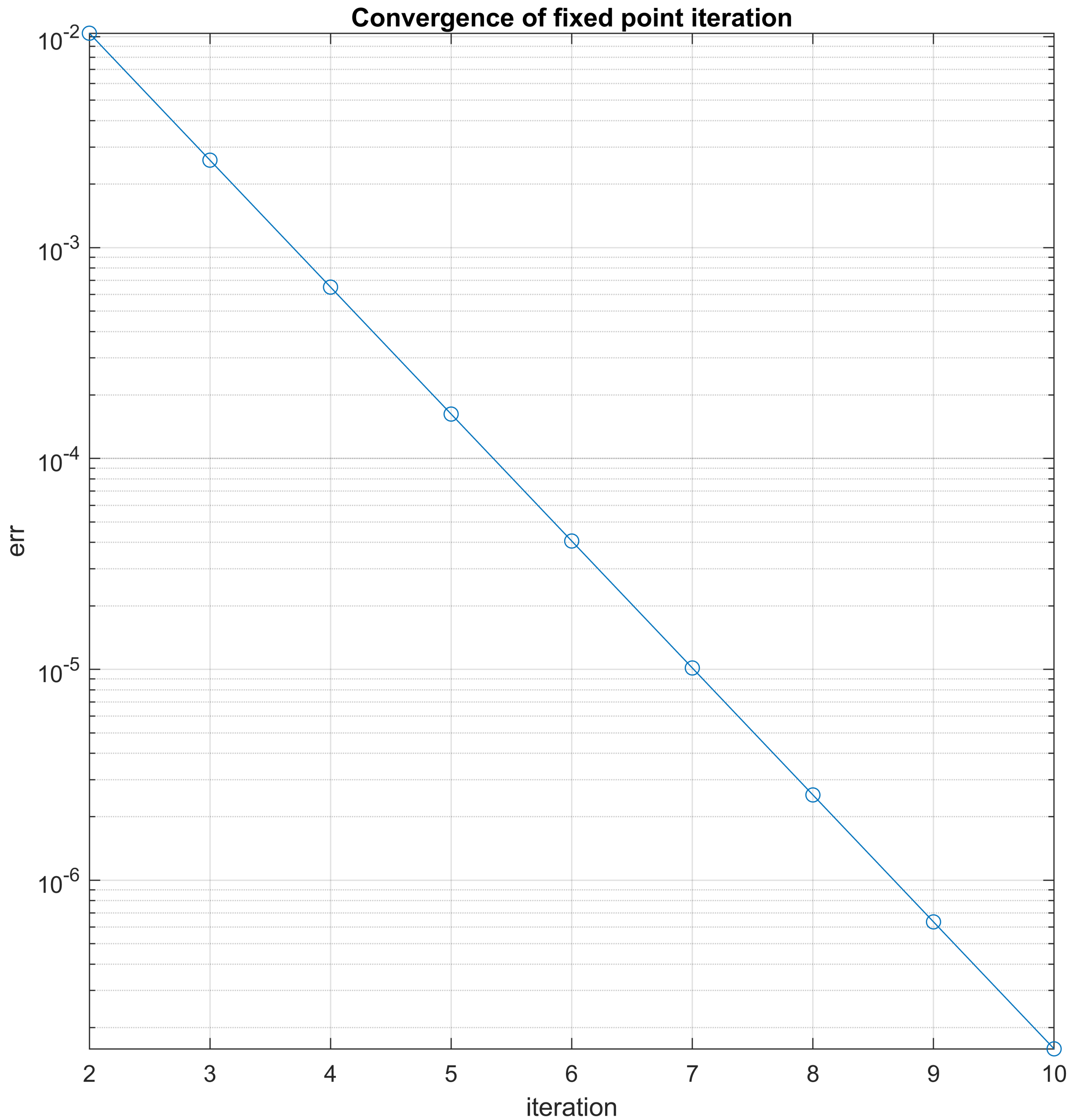



```
%The iteration converges very fast and the errors after the fourth are just
%zeros and becomes invisible to semilogy which is evident in the fact that
%the analytical calculation is super linear convergence as it is 0
%This is means that our sigma will also have to start at a lower value to
%ensure that we do not have a 0/0 when approximating the value
ApproximateValue = err(3:5)./err(2:4)
```

```
ApproximateValue = 3×1
        0.00507614213198006
       2.60301424037639e-05
                          0
```

```
%We keep it below 5 to ensure no 0/0 calculations but based off of the
%values the approximate error appears to be going towards zero which based
%on the analytical calculations done above we can confirm that this is
%approximately estimating the rate sigma
```

```
h = @(x) pi + (1/4)*sin(x);
n = 10;
x = zeros(n, 1);
err = zeros(n,1);
r = pi;
x(1) = 3.1;
for k = 1:n-1
    x(k+1) = h(x(k));     %FPI
    err(k+1) = abs(x(k+1) - r);
end
clf
semilogy(err, 'o-')
xlabel('iteration')
ylabel('err')
title('Convergence of fixed point iteration')
axis tight, grid on
```



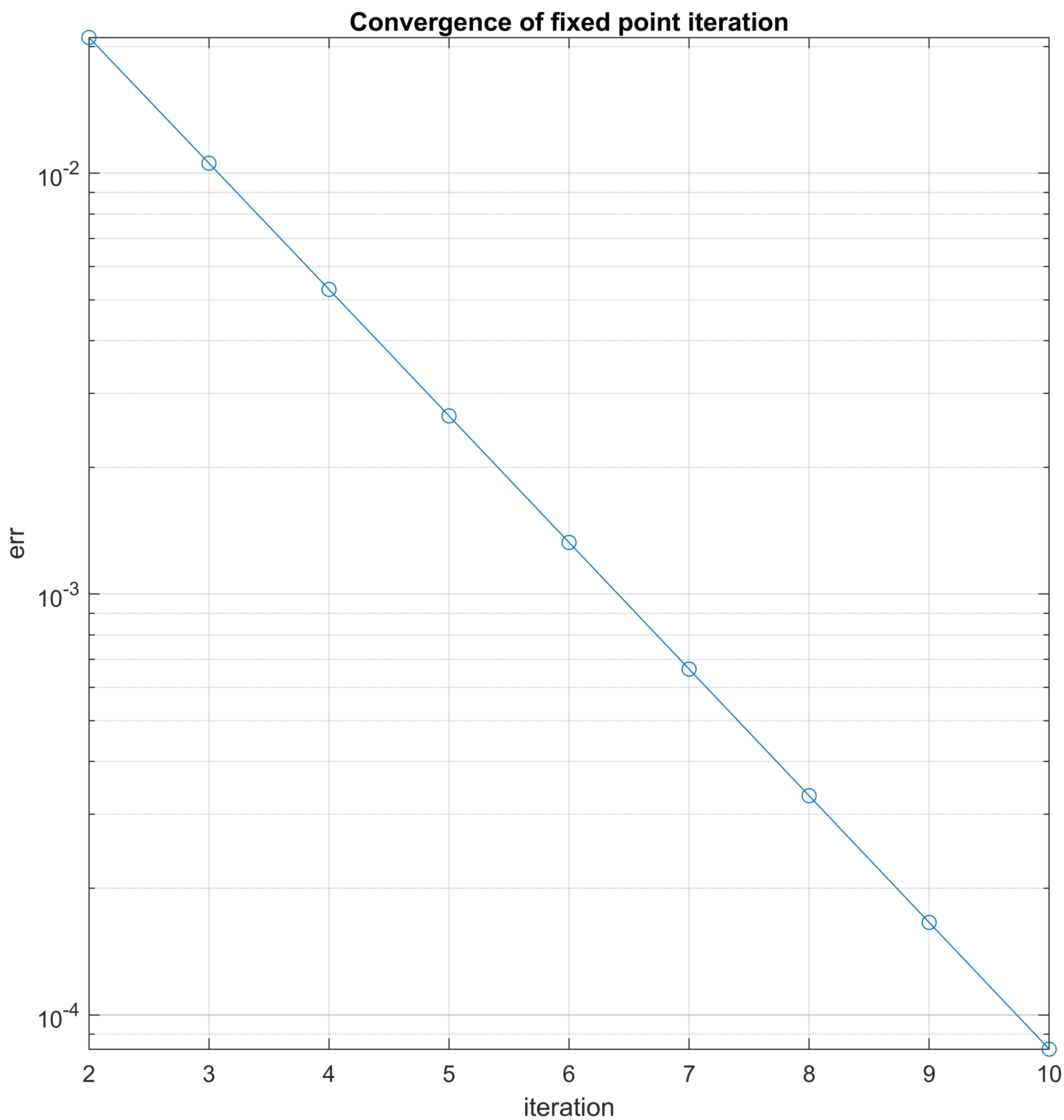

```
ApproximateValue = err(4:10) ./ err(3:9)
```

```
ApproximateValue = 7×1
    0.24999971860531
   0.249999982413071
   0.249999998901543
    0.24999999993438
                0.25
   0.249999999912506
                0.25
```

```
%Based on our analytical calculations above we should see the approximate
%rate to be close to 1/4 which based on the error approximations we see
```

```
%here that is confirmed as all the values appear to be right around 1/4
%with a few discrepancies here and there
```

```
i = @(x) x + 1 - tan(x/4);
n = 10;
x = zeros(n, 1);
err = zeros(n,1);
r = pi;
x(1) = 3.1;
for k = 1:n-1
    x(k+1) = i(x(k));     %FPI
    err(k+1) = abs(x(k+1) - r);
end
clf
semilogy(err, 'o-')
xlabel('iteration')
ylabel('err')
title('Convergence of fixed point iteration')
axis tight, grid on
```




```
ApproximateValue = err(4:10) ./ err(3:9)
```

```
ApproximateValue = 7×1
         0.501315318857684
         0.500660545264199
         0.500331000135971
         0.500165682707762
           0.5000828871099
         0.500041455004945
         0.500020730367182
```

```
%Based on the analytical calculation above we should see the approximate
%rate to be close to 1/2 which in our error calculations we do see that to
```

```
%be the case with many of the values being very close to 1/2 with only a
%few discrepancies here and there
```

Problem 5.

5. (Convergence of Newton's method) Answer the following questions *by hand*, without using MATLAB.

   (a) Discuss what happens when Newton's method is applied to find a root of

$$f(x) = \text{sign}(x)\sqrt{|x|},$$

   starting at $x_0 \neq 0$. [1]

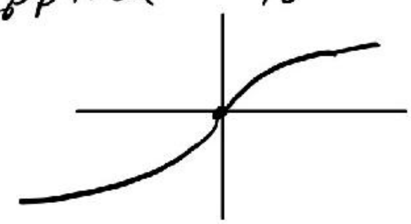
[1] sign(x) is 1 if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$.

(b) In the case of a multiple root, where $f(r) = f'(r) = 0$, the derivation of the quadratic error convergence is invalid. Redo the derivation to show that in this circumstance and with $f''(r) \neq 0$ the error converges only linearly.

5a.) Discuss what happens when Newton's method is applied to find a root of

$$f(x) = \text{sign}(x)\sqrt{|x|}$$

Starting at $x_0 \neq 0$.



$\text{sign}(x) = 1$ if $x > 0$

$\text{sign}(x) = -1$ if $x < 0$

$\text{sign}(x) = 0$ if $x = 0$

$$f(x) = \text{sign}(x)\sqrt{|x|} = \begin{cases} \sqrt{x}, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -\sqrt{-x}, & \text{if } x < 0 \end{cases}$$

Case: $x > 0$

$$f'(x) = \frac{d}{dx}\sqrt{x}$$

$$= \frac{1}{2}x^{-\frac{1}{2}}$$

$$= \frac{1}{2\sqrt{x}}$$

Case: $x < 0$

$$f'(x) = \frac{d}{dx}\left(-\sqrt{-x}\right)$$

$$= \frac{1}{2}(-x)^{-1/2}$$

$$= \frac{1}{2\sqrt{-x}}$$

Use Newton's iteration formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Case 1: $x>0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$= x_0 - \frac{\sqrt{x_0}}{\frac{1}{2\sqrt{x_0}}}$$

$$= x_0 - 2(\sqrt{x_0}\cdot\sqrt{x_0})$$

$$= x_0 - 2x_0$$

$$x_1 = -x_0$$

Case 2: $x<0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_1 = x_0 - \frac{-\sqrt{-x_0}}{\frac{1}{2\sqrt{-x_0}}}$$

$$= x_0 - -2\sqrt{-x_0}\cdot\sqrt{-x_0}$$

$$x_1 = -x_0$$

When newtons formula is applied to the equation we see that in both cases the values become the negative value of the previous step. As we see from the graph above our guess would be around 0 however we can't confirm this and newton's method shows that it has one root that does not need many approximations to reach the root.

b.) We are under the assumption that $f(r) = f'(r) = 0$ while $f''(r) \neq 0$ so we will use Taylor series expansion to see this result.

$$\epsilon_{k+1} = \epsilon_k - \frac{f(r+\epsilon_k)}{f'(r+\epsilon_k)}$$

$$= \epsilon_k - \frac{\cancel{f(r)} + \cancel{\epsilon_k f'(r)} + \frac{1}{2}\epsilon_k^2 f''(r) + \frac{1}{3!}\epsilon_k^3 f'''(r) + O(\epsilon_k^4)}{\cancel{f'(r)} + \epsilon_k f''(r) + \epsilon_k^2 f'''(r) + O(\epsilon_k^3)}$$

$$= \epsilon_k - \frac{\frac{1}{2}\epsilon_k^2 f''(r) + \frac{1}{3!}\epsilon_k^3 f'''(r) + O(\epsilon_k^4)}{\epsilon_k f''(r) + \epsilon_k^2 f'''(r) + O(\epsilon_k^3)}$$

$$= \epsilon_k - \frac{\epsilon_k^{\cancel{2}} \cancel{f''(r)}\left[\frac{1}{2!} + \frac{1}{3!}\epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]}{\cancel{\epsilon_k}\cancel{f''(r)}\left[1 + \epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]}$$

$$= \epsilon_k - \frac{\epsilon_k\left[\frac{1}{2!} + \frac{1}{3!}\epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]}{\left[1 + \epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]}$$

$$\frac{1}{1 + \epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)} = 1 - \epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)$$

treat this as $-\alpha$ for a geometric series

$$\frac{1}{1-\alpha} = \sum_{k=0}^{\infty} \alpha^k = 1 + \alpha + \alpha^2 + \dots$$

$$\Rightarrow \epsilon_k - \epsilon_k\left[\frac{1}{2} + \frac{1}{3!}\epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]\left[1 - \epsilon_k \frac{f'''(r)}{f''(r)} + O(\epsilon_k^2)\right]$$

$$= \epsilon_k - \epsilon_k\left[\frac{1}{2} + \left(\frac{1}{6} - 1\right)\frac{f'''(r)}{f''(r)}\epsilon_k + O(\epsilon_k^2)\right]$$

$$= \boxed{\frac{1}{2}\epsilon_k + \frac{5}{6}\frac{f'''(r)}{f''(r)}\epsilon_k^2 + O(\epsilon_k^3)}$$

Function Lambert's W Function

```
function y = LambertW(x)
% Function Lambert W Function
% Calculates the inverse of inputted x vector to solve for another vector
% y
f = @(x,y) x - y*exp(y);
y = zeros(size(x));
for j = 1:length(x)
    y(j) = fzero(@(y) f(x(j), y), (-1 + sqrt(1 + 4*x(j)))/2 );
end
end
```