TASK
Implement estimation of the following non-linear regression model using a fast and stable algorithm:

\[ Y = \text{Beta}_1 \times (1 - 1 / \sqrt{1 + 2 \times \text{Beta}_2 \times X}) + \text{Epsilon}. \]

SOLUTION
The Matlab code is displayed below.

```matlab
function [Beta, RSS, SigmaSq, CovarianceOfBeta, StandardErrors, Iter] = NonLinearRegression(Y,X,InitialBeta)

% This function implements an iterative procedure for estimating a
% non-linear regression model of the form:
% \[ Y = h(X,\text{Beta}) + \text{Epsilon}, \] (*)
% where X and Y are random variables, \( \text{Beta} = (\text{Beta}_1, \text{Beta}_2) \) and
% \[ h(X,\text{Beta}) = \text{Beta}_1 \times (1 - 1 / \sqrt{1 + 2 \times \text{Beta}_2 \times X}). \]
% The iterative procedure is based on LINEARIZATION of equation (*) by
% applying Taylor expansion to function \( h(X,\text{Beta}) \) and running Ordinary
% Least Squares (OLS) for the resulting linear regression. Here is how
% linearization works:
% 1) Choose reasonable possible value for \( \text{Beta} = (\text{Beta}_1, \text{Beta}_2) \). This will
% be our \( \text{Beta}_0 \).
% 2) Treat \( h(X,\text{Beta}) \) as a function of \( \text{Beta} \) and write down a first-order
% Taylor approximation for \( h(X,\text{Beta}) \) in the neighborhood of \( \text{Beta}_0 \).
% Substitute the approximation into equation (*). Now the right hand side
% is a linear function of \( \text{Beta} \). Therefore \( \text{Beta} \) can be estimated using methods
% for linear regression.
% 3) Estimate \( \text{Beta} \) by OLS. Let \( \text{Beta} \_\text{Hat} \) be our estimate.
% 4) If \( \text{Beta} \_\text{Hat} \) is within tolerance from \( \text{Beta}_0 \), choose \( \text{Beta} \_\text{Hat} \) as the
% final estimate of coefficients \( \text{Beta} \). Otherwise, set \( \text{Beta}_0 \) to \( \text{Beta} \_\text{Hat} \)
% and go to the step 2.
% The substantiation of this iterative procedure is well explained in
% Greene, "Econometric Analysis", Ch. "Nonlinear Regression Models".

% This ensures that X and Y are column vectors.
Size    = size(Y);
if Size(2) > 1
    Y = Y';
end
Size    = size(X);
if Size(2) > 1
    X = X';
end
```

% Initialization.
Tolerance   = 1e-6;
Discrepancy = 100;
Iter        = 0;
Beta        = InitialBeta';

while Discrepancy > Tolerance
    % Update the number of iterations.
    Iter    = Iter+1;
    
    % hfun evaluated at the initial values:
    h0      = Beta(1) * (1 - 1 ./ sqrt( 1 + 2*Beta(2)*X ) );
    
    % Need pseudoregressors at the initial values
    x1_0    = 1 - 1 ./ sqrt( 1 + 2*Beta(2)*X );
    x2_0    = Beta(1)*X .* ( 1 + 2*Beta(2)*X ).^(-3/2);
    
    % Collect them in our pseudoregressors matrix
    X0          = [x1_0 x2_0];
    
    % Now compute X0Beta0
    h0_delta    = X0*Beta;
    
    % We're now ready to compute the new dependent variable
    y0          = Y - h0 + h0_delta;
    
    % We can estimate our linearized model with OLS
    NewBeta     = regress(y0,X0);
    
    % Check improvement
    Discrepancy = max(abs((Beta-NewBeta) ./ InitialBeta'));
    
    % Update the coefficients.
    Beta        = NewBeta;
end

% Calculating various statistics to be able to produce the covariance
% matrix of the coefficients estimates.
N       = length(Y);
K       = 2;
Fit     = Beta(1) * (1 - 1 ./ sqrt( 1 + 2*Beta(2)*X ) );
RSS     = sum( (Y - Fit).^2 );
SigmaSq = 1/(N - K) * RSS;

% Need pseudoregressors at the initial values
x1_0    = 1 - 1 ./ sqrt( 1 + 2*Beta(2)*X );
X0      = [x1_0 x2_0];

CovarianceOfBeta    = inv(X0'*X0) * SigmaSq;
StandardErrors      = [sqrt(CovarianceOfBeta(1,1)),
                      sqrt(CovarianceOfBeta(2,2))];