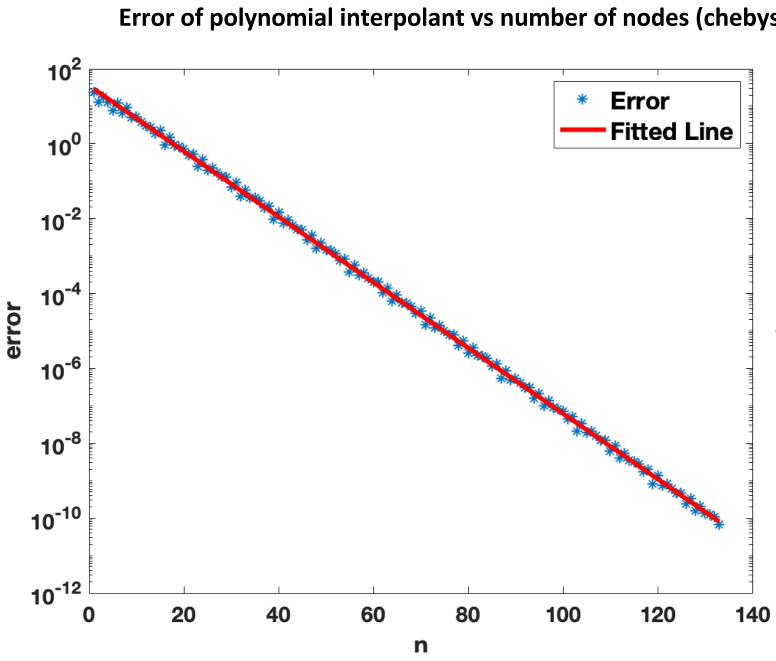
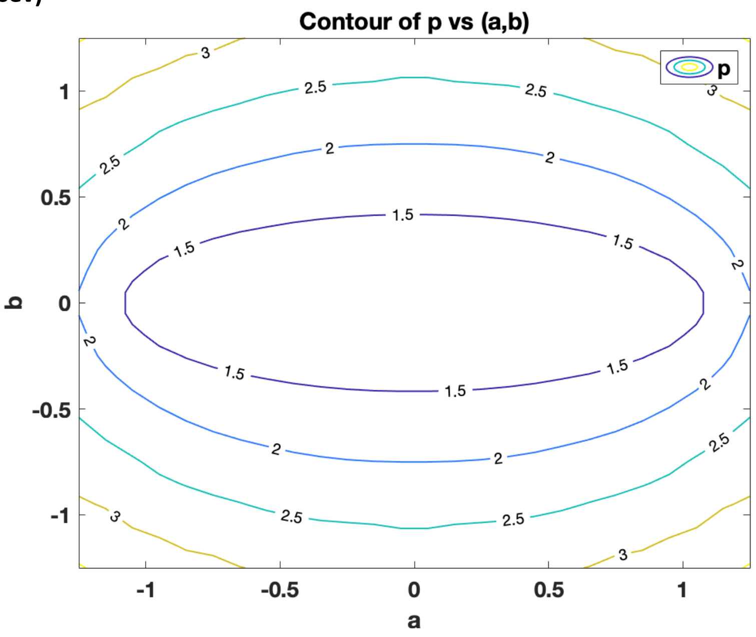
Assignment 5 – Ngawang Kyirong 301312227





b.) ρ can be found by using 2 consecutive error values approximated by the line of best fit. For example: at n = 1, y = 29.0607 & n = 2, y = 23.7555. Then we can manipulate the given equations as shown: $error_n \leq C\rho^{-n}$ and $error_{n+1} \leq C\rho^{-(n+1)}$ => $\frac{error_n}{error_{n+1}} \leq \frac{C\rho^{-n}}{C\rho^{-(n+1)}}$ => $\frac{error_n}{error_{n+1}} \leq \frac{1}{\rho^{-1}} \leq \rho.$ Using the values from x = 1 and x = 2, it is easily shown that $\rho \approx 1.2233$ for (a, b) = (0.2, 0.2). Furthermore, for (a, b) = (0.5, 0.5) => $\rho \approx 1.6952$ and for (a, b) = (0.7, 1) => $\rho \approx 2.6410$.

c.) The code uses a tolerance level for the error instead of n because if you proceed with larger values of n needed, the results vary depending on (a,b) and the error starts to increase after a certain point. Furthermore, the error stops decreasing and upon zooming in on the area where the error increases, the error is unpredictable relative to each other and not linear to each other thus changing our value of p (thus changing the bound) each further iteration.

e.) As a and b increases together the value of p also increases. Therefore, the level curves are further away from the center. As a and b decreases together the value of p decreases. Therefore, the level curves are closer to the center. However, for large values of b and smaller values of a, b seems to have more effect than a and it converges faster to its minimum error and starts to rise again. Moreover, for large values of a and smaller values of b, it converges slower to its minimum error point. This can also be seen through the contour plot. Any large value of b remains on the outer ends of the level curves (p is relatively large). Contrastingly, large values of a can still hit small values of p depending on the value of b.

```matlab
err_grid = (linspace(-1,1,10000))'; % Equally-spaced grid of 10000 nodes to compute error on
tol = 1e-10; % tolerance used to determined largest value of n

%a = 0.7; b = 1; % Set values of a,b
a = -1.25:.1:1.25;
b = -1.25:.1:1.25;
z = zeros(length(a), length(a));
%f = @(x) 1./((x-a).^2+b.^2); % Define f(x)

%f_grid = f(err_grid); % Evaluate f on the error grid
%errvals = []; % Initialize an error vector
%err = 1; % set the error initially to 1
%n = 0; % initialize n


for i = 1:length(a)
    for j = 1:length(a)

        f = @(x) 1./((x-abs(a(j))).^2+abs(b(i)).^2); % Define f(x)
        f_grid = f(err_grid);
        errvals = []; % Initialize an error vector
        err = 1; % set the error initially to 1
        n = 0; % initialize n

        while err > tol
            n = n+1;

            % compute points
            x = (linspace(0,1,n+1))';
            x = cos(pi*x);

            % compute weights
            w = (-1).^((0:n)');
            w(1) = 1/2; w(n+1) = w(n+1)/2;

            y = f(x); % Evaluate f at the nodes
            p = baryinterp(x,w,y,err_grid); % Compute P(x) on the error grid
            err = max(abs(p - f_grid)); % Compute error
            errvals = [errvals err]; % Update error vector

        end
        logerr = log(errvals);
        coeff = polyfit(1:n,logerr, 1);
        vals = polyval(coeff, 1:n);
        evals = exp(vals);
        p = evals(2)/evals(3);
        z(i,j) = p;
    end
end

%cont = [a,b,z];
contour(a,b,z,'ShowText','on','LineWidth', 1)

% produce line of best fit
%{
logerr = log(errvals);
coeff = polyfit(1:n,logerr, 1)
vals = polyval(coeff, 1:n);

figure(1);
semilogy(1:n,errvals,'*');
hold on
semilogy(1:n, exp(vals), 'r','linewidth',3); % plot the line of best fit
hold off

set(gca,'FontSize',14);
xlabel('n','fontsize',16);
ylabel('error','fontsize',16);
legend({'Error','Fitted Line'},'fontsize', 16, 'Location','northeast')
%}
```