Assignment 2

Gudjon Einar Magnusson

October 7, 2016

1

1.a

Figure 1 shows the function $u(t) = 1/(1.1 - cost)$ and the polynomial interpolation $p(t)$ using 11 equidistant points.

The interpolation is performed using the functions *cos_transform* and *inv_cos_transform*, shown below.

```
function a = cos_transform( U )
    n = length(U)-1;
    U_extended = [U; flip(U(2:end-1))];         % Extend U by mirroring. U is even and periotic

    c = fft(U_extended)./length(U_extended);
    a = [c(1); c(2:n)+c(end:-1:n+2); c(n+1)];   % cosine coefficients a_0,...,a_n
end
```

```
function U = inv_cos_transform( a )
    n = length(a)-1;
    b = zeros(length(a)-2, 1);
    c = [a(1); a(2:end)/2; zeros(n-1,1)] + [zeros(n,1); a(end:-1:2)/2] + ...
         [0; b/2i; 0; -b(end:-1:1)/2i];

    U = ifft(c)*length(c);
    U = U(1:length(U)/2);       % remove extended part of the function
end
```

1.b

Figure 2 shows how the error $\|u - p_n\|_\infty$ decays as n increases. The Log of the error drops linearly which indicates exponential convergence.

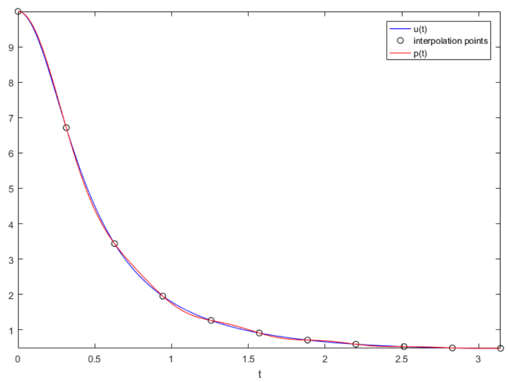

Figure 1: $u(t)$ and the interpolating polynomial $p(t) \in \mathcal{P}_{10}$. Using 11 equidistant nodes.

Below you can see the output produced when estimating the error for $n = 2, 4, \ldots, 50$. N is the order of the polynomial. E is the error estimate E_n. S is the slope between the points $(n-2, logE_{n-2}), (n, logE_n)$.

```
N: 4   E: 1.6581      S: -0.91944
N: 6   E: 0.60969     S: -1.0005
N: 8   E: 0.2606      S: -0.84995
N: 10  E: 0.11235     S: -0.84139
N: 12  E: 0.046269    S: -0.88712
N: 14  E: 0.018576    S: -0.91262
N: 16  E: 0.0078078   S: -0.86673
N: 18  E: 0.0032452   S: -0.87796
N: 20  E: 0.001331    S: -0.89126
N: 22  E: 0.00054384  S: -0.89501
N: 24  E: 0.00022618  S: -0.87733
N: 26  E: 9.3312e-05  S: -0.88537
N: 28  E: 3.8287e-05  S: -0.89084
N: 30  E: 1.5764e-05  S: -0.8874
N: 32  E: 6.5138e-06  S: -0.88379
N: 34  E: 2.6821e-06  S: -0.88731
N: 36  E: 1.1014e-06  S: -0.89004
N: 38  E: 4.5448e-07  S: -0.88516
N: 40  E: 1.8723e-07  S: -0.8868
N: 42  E: 7.7184e-08  S: -0.88617
N: 44  E: 3.1691e-08  S: -0.89016
N: 46  E: 1.3085e-08  S: -0.88456
N: 48  E: 5.3855e-09  S: -0.88774
N: 50  E: 2.2195e-09  S: -0.88644
```

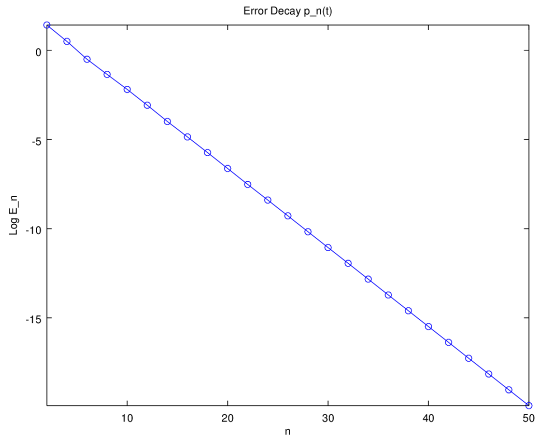

Figure 2: The error $\|u - p_n\|_\infty$ drops exponentially as n increases.

1.c

Derp

1.d

Figure 3 shows the function $v(x) = |x|^{1/3}$ and the polynomial interpolation $p(t)$ using 9 Chebyshev nodes.

2

2.a

The function *interperr_eq* approximates the function $u(x) = (1 + 30x^2)^{-1}$ with the interpolating polynomial $p_n(x)$ using $n+1$ equidistant nodes. Using Lagrange formula with equidistant nodes suffers from Runge's phenomenon. It oscillates far from the true value at the edges of the range.

Figure 4 shows the error $\|u - p_n\|_\infty$ goes down at first but then increases exponentially as n increases. The oscillation get worse and worse as n increases.

To roughly estimate the decay rate α, I divided the range of E with the range of N to get a slope. Figure 4 shows he line $logCe^{\alpha n}$ using my

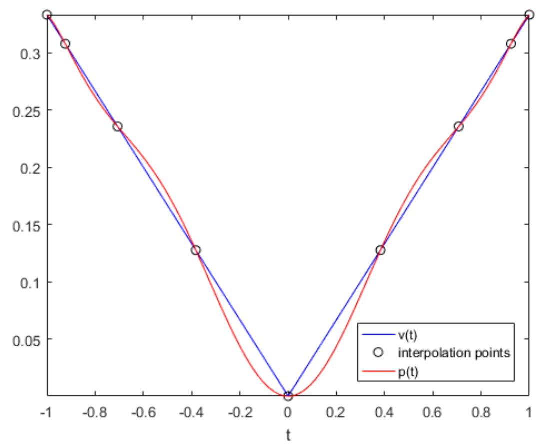

Figure 3: $v(x)$ and the interpolating polynomial $p(x) \in \mathcal{P}_8$. Using 9 Chebyshev nodes.

estimate for α and it does satisfy the the condition $E_n \leq Ce^{\alpha n}$. In this case $\alpha = 0.29310$ and $C = 0.4$.

```
function err = interperr_eq( n )
    np = 1000;                        % Number of points to plot with
    u = @(x) (1./(1+30*x.^2));        % function u(x)

    x = -1 + (0:n)' *2/n;
    xp = -1 + (0:np-1)' *2/np;

    U = u(x);                         % Evaluate u(x)
    Px = modlagr(x, U, xp);
    Ux = u(xp);

    err = max(abs(Px - Ux));
end
```

2.b

The function *interperr_ch* approximates the function $u(x) = (1 + 30x^2)^{-1}$ with the interpolating polynomial $p_n(x)$ using $n + 1$ Chebyshev nodes. By using Chebyshev nodes the oscillation issue is fixed. Nodes a placed closer together at the edges of the range.

Figure 5 shows the error $\|u - p_n\|_\infty$ decreases exponentially as n increases. It also shows my rough estimate for α as the line $logCe^{\alpha n}$, it does satisfy the

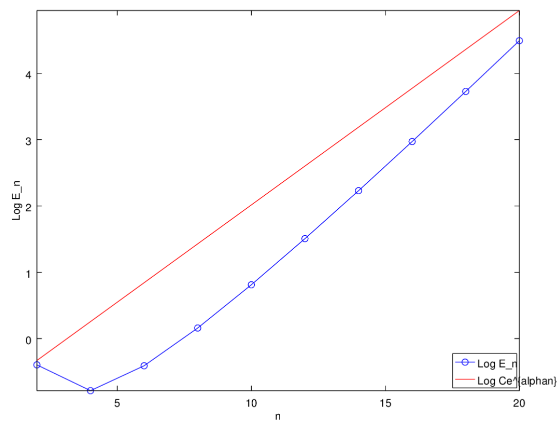

Figure 4: $LogE_n$ and a rough estimate of the decay rate α

the condition $E_n \leq Ce^{\alpha n}$. In this case $\alpha = -0.18374$ and $C = 1.04$.

```
function err = interperr_ch( n )
    np = 1000;                        % Number of points to plot with
    u = @(x) (1./(1+30*x.^2));        % function u(x)

    t = (0:n)' *pi/n;
    tp = (0:np-1)' *pi/np;
    x = cos(t);
    xp = cos(tp);

    U = u(x);                         % Evaluate u(x)
    Px = modlagr(x, U, xp);
    Ux = u(xp);

    err = max(abs(Px - Ux));
end
```

2.c

3

3.a

3.b

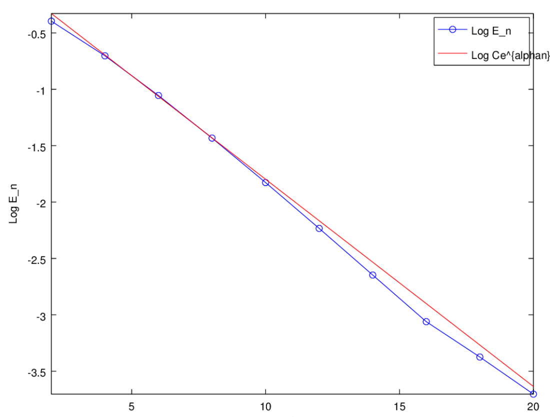

Figure 5: $LogE_n$ and a rough estimate of the decay rate α