EPFL

Numerical Analysis and Computational Mathematics

Fall Semester 2025 – CSE Section

Prof. Laura Grigori

Assistant: Israa Fakih

Session 6 – October 15, 2025

Solutions – Approximation of functions and data

Solution I (MATLAB)

a) We use the following MATLAB commands to obtain the polynomial $\Pi_n f(x)$ of degree $n = 9$ interpolating $f(x)$ at $n+1$ uniformly spaced nodes over $[a, b] = [0, 1]$ and the least-squares approximating polynomial $\widetilde{f}_m(x)$ of degree $m = 2$. The results are reported in Figure 1 (left).

```
a = 0; b = 1;
g = @(x) 10 * x.^2;
f = @(x) g(x) + 2 * rand(size(x))-1;
n = 9;
x_nodes = linspace(a,b,n+1);
y_nodes = f(x_nodes);

x_values = linspace(0,1,1001);
f_values = f(x_values);
g_values = g(x_values);

Pinterp = polyfit( x_nodes, y_nodes, n );
Pinterp_values = polyval( Pinterp, x_values );

PLeastSquares = polyfit( x_nodes, y_nodes, 2 );
PLeastSquares_values = polyval( PLeastSquares, x_values );

plot( x_values, f_values, '-g', x_values, g_values, '-k', ...
        x_values, Pinterp_values, '-r', x_values, PLeastSquares_values, '-b' )
legend( {'$f(x)$', '$g(x)$', '$\Pi_n f(x)$', '$\tilde{f}_2(x)$'},...
    'Interpreter','latex');
```

As we can observe in Figure 1 (left), the interpolating polynomial $\Pi_9 f(x)$ is inadequate for representing the behavior of the function $g(x)$ on the interval I since it interpolates the data points $\{(x_i, f(x_i))\}$, which take the noise ε into account. Conversely, the least-squares approximating polynomial $\widetilde{f}_2(x)$ is suitably describing the function $g(x)$ on the interval I, when constructed from the same data points. We observe that $\widetilde{f}_2(x)$ is not interpolatory, rather it represents the quadratic polynomial $\widetilde{f}_2(x) = a_0 + a_1 x + a_2 x^2$ that minimizes the

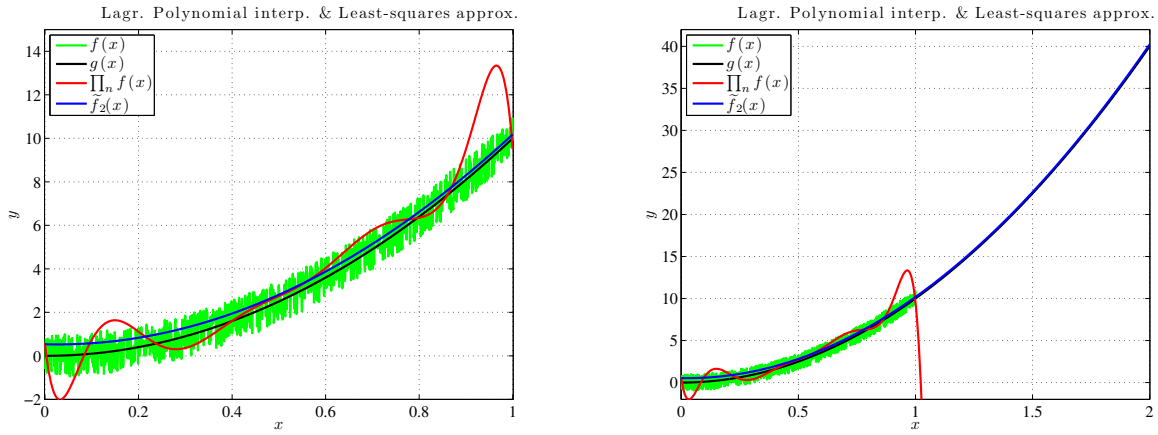

Figure 1: Polynomial $\Pi_n f(x)$ of degree $n = 9$ interpolating $f(x)$ (red) and least-squares approximating polynomial $\widetilde{f}_m(x)$ of degree $m = 2$ (blue) for $x \in I = [0, 1]$ (left) and extrapolated for $x \in [0, 2]$ (right); comparison with $f(x)$ (green) and $g(x)$ (black).

discrete interpolation error $\Phi(a_0, a_1, a_2) = \sum_{i=0}^{n} \left(f(x_i) - (a_0 + a_1 x_i + a_2 x_i^2)\right)^2$. In this case we obtain $a_0 = 0.5337$, $a_1 = -0.5480$, and $a_2 = 10.1978$ (see the variable `PLeastSquares`). We deduce that $\widetilde{f}_2(x)$ is a suitable approximation of the signal $g(x) = 10x^2$.

We remark that the values obtained vary depending on the output of the MATLAB function `rand`.

b) We extrapolate the interpolating polynomial $\Pi_9 f(x)$ and the least-squares approximating polynomial $\widetilde{f}_2(x)$ outside the interval I, specifically for $x \in [1, 2]$. We report the results in Figure 1 (right) as obtained by the following commands.

```
x_values2 = linspace( 0, 2, 1001 );
g_values2 = g(x_values2);
Pinterp_values2 = polyval( Pinterp, x_values2 );
PLeastSquares_values2 = polyval( PLeastSquares, x_values2 );
Pinterp_values2_x2 = Pinterp_values2( end )
%  Pinterp_values2_x2 =
%    -3.3843e+06
PLeastSquares_values2_x2 = PLeastSquares_values2(end)
%  PLeastSquares_values2_x2 =
%     40.2288
plot( x_values, f_values, '-g', x_values2, g_values2, '-k', ...
     x_values2, Pinterp_values2, '-r', x_values2, PLeastSquares_values2, '-b'  )
legend( {'$f(x)$', '$g(x)$', '$\Pi_n f(x)$', '$\tilde f_2(x)$'}, ...
    'Interpreter','latex');
```

We observe that the interpolating polynomial $\Pi_9 f(x)$ is inadequate for representing the behavior of the function $g(x)$, and eventually $f(x)$, outside the interval I, as confirmed by the value assumed at $x = 2$, e.g. $\Pi_9 f(2) = -3.3843 \cdot 10^6$. Conversely, the least-squares approximating polynomial $\widetilde{f}_2(x)$ still adequately represents the function $g(x)$ outside the interval I and allows to extrapolate the values of the function $g(x)$. For example, we obtain that $\widetilde{f}_2(2) = 40.2288$ $(g(2) = 40)$.

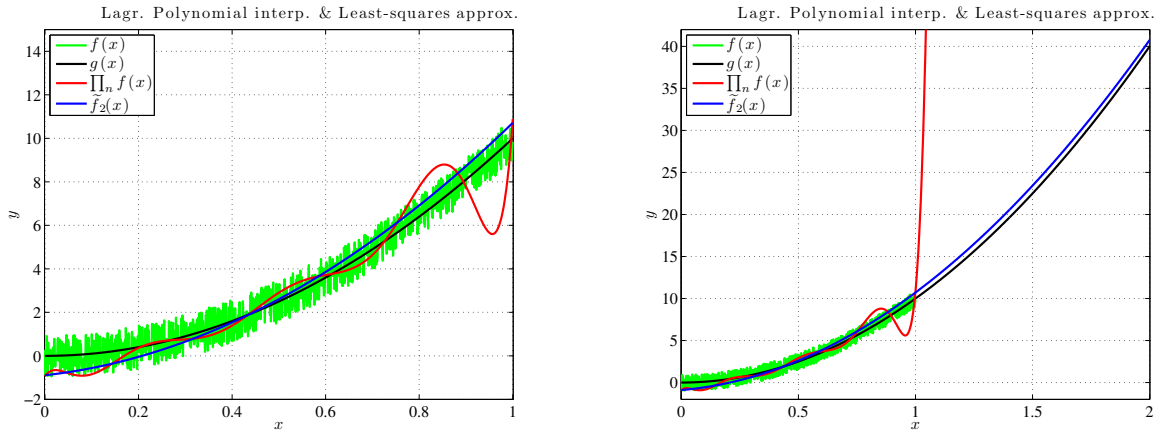

Figure 2: Polynomial $\Pi_n f(x)$ of degree $n = 9$ interpolating $f(x)$ (red) and least-squares approximating polynomial $\widetilde{f}_m(x)$ of degree $m = 2$ (blue) for $x \in I = [0, 1]$ (left) and extrapolated for $x \in [0, 2]$ (right); comparison with $f(x)$ (green) and $g(x)$ (black) for a noise $\varepsilon(x)$ different than the one considered in Figure 1.

c) We repeat points a) and b). We obtain the results reported in Figure 2. As we can observe, the least-squares approximating polynomial still provides an adequate representation of the function $g(x)$ both inside and outside the interval I, with limited sensitivity with respect to the new data points $\{(x_i, f(x_i))\}_{i=0}^{n}$. On the other hand, the interpolating polynomial still does not represent the function properly on I, and yields incorrect extrapolated values. Moreover, the interpolating polynomial exhibits a strong sensitivity to the noise $\varepsilon(x)$ as confirmed by comparing Figures 1 and 2.

Solution II (MATLAB)

a) We use the following commands in MATLAB to obtain the piecewise linear interpolants $\Pi_1^H f(x)$ reported in Figure 3 for $n = 2, 7, 12$, and 22 sub-intervals.

```
f = @(x) exp(-x.^2 / 2);
a = -5; b = 5;
x_values = linspace( a, b, 1001 );
f_values = f( x_values );
n_vect = [2 7 12 22 27 32 ];
for n = n_vect
   x_nodes = linspace( a, b, n + 1 );
   y_nodes = f( x_nodes );
   yp1_values = interp1( x_nodes, y_nodes, x_values );
   figure
   plot( x_values, f_values, '--k', x_values, yp1_values, '-k', ...
           x_nodes, y_nodes, 'xk');
   legend( 'f(x)', '\Pi_1^H f(x)', '(x_i,f(x_i))');
   pause(0.1)
```

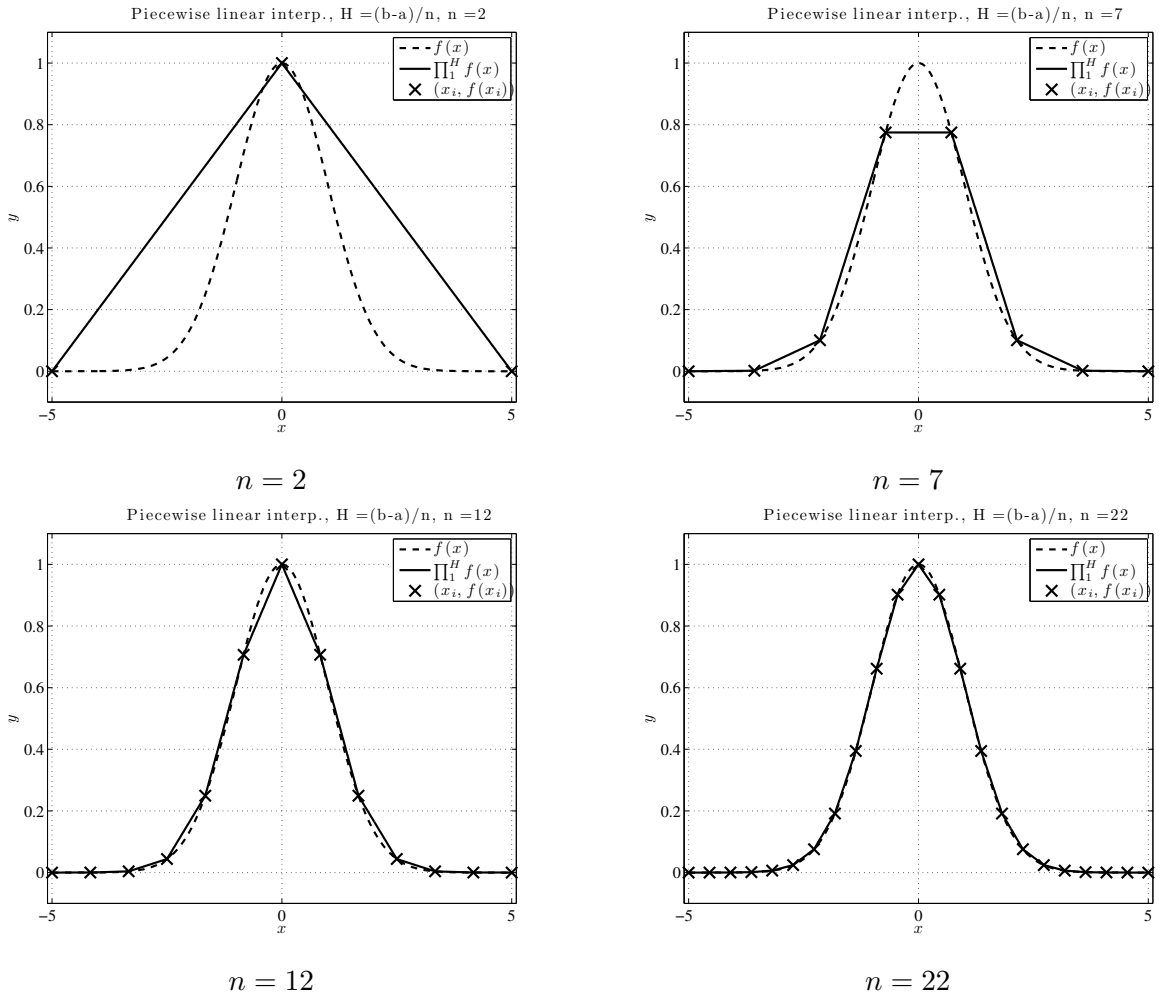

Figure 3: Piecewise linear interpolants $\Pi_1^H f(x)$ for $n = 2, 7, 12$, and 22 sub-intervals.

b) We report in Figure 4 the errors $e_1^H(f)$ corresponding to $\Pi_1^H(f)$ vs. n.

```
n_vect = 2:32;
err_yp1 = [];
for n = n_vect
   x_nodes = linspace( a, b, n + 1 );
   y_nodes = f( x_nodes );
   yp1_values = interp1( x_nodes, y_nodes, x_values );
   err_yp1 = [ err_yp1, max( abs( f_values - yp1_values ) ) ];
end
```

We observe that the errors $e_1^H(f)$ converge to zero as the number of sub-intervals n increases. The oscillatory behavior is due to the positioning of the nodes for even and odd values of n. In particular, we observe that, when n is odd, the maximum error $e_1^H(f)$ is obtained at $x = 0$ and it is larger than the error corresponding to $n - 1$ sub-intervals (in which case, a node is placed at $x = 0$).

c) We use the following commands to obtain the spline $s_3(x)$ reported in Figure 5.

```
n = 7;
x_nodes = linspace( a, b, n + 1 );
y_nodes = f( x_nodes );
yp1_values = interp1( x_nodes, y_nodes, x_values );
s3_values = spline( x_nodes, y_nodes, x_values );
figure;
plot( x_values, f_values, '--k', x_values, yp1_values, '-k', ...
       x_values, s3_values, '-r', x_nodes, y_nodes, 'xk' );
```

We observe the smoothness of the spline $s_3(x)$, which is globally C^2-continuous on the interval I: $s_3 \in C^2(I)$ and $s_3 \in C^\infty((x_i, x_{i+1}))$ for $i = 0, \ldots, n-1$.

By using the following commands, we show in Figure 6 the errors $e_{s_3}(f)$ vs. the number of sub-intervals n. We observe the convergence of the error to zero as n increases.

```
n_vect = 2:32;
err_s3 = [];
for n = n_vect
   x_nodes = linspace( a, b, n + 1 );
   y_nodes = f( x_nodes );
   s3_values = spline( x_nodes, y_nodes, x_values );
   err_s3 = [ err_s3, max( abs( f_values - s3_values ) ) ];
end
figure; semilogy( n_vect, err_s3, '-ok' );
```

Solution III (Theoretical)

a) Let $f \in C^2(I)$, with $I = [a, b]$. The maximum error associated to the piecewise linear interpolant $\Pi_1^H f(x)$ on I, i.e. $e_1^H(f) := \max_{x \in I} \left| f(x) - \Pi_1^H f(x) \right|$, can be bounded by an error

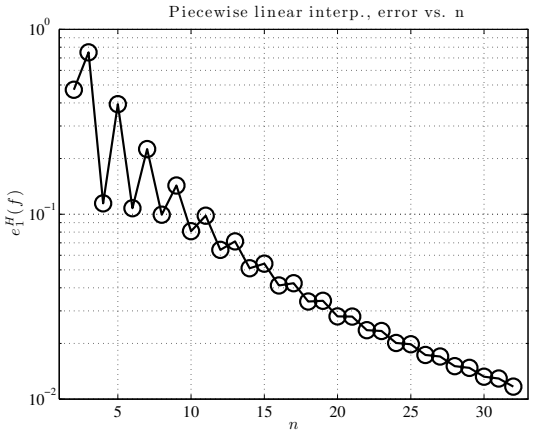

Figure 4: Errors $e_1^H(f)$ vs. n.

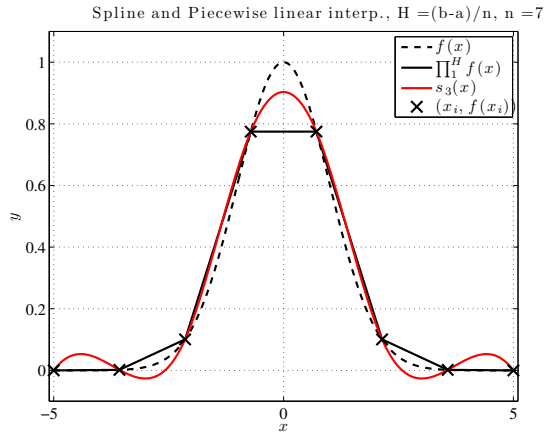

Figure 5: Spline $s_3(x)$ for $n = 7$ sub-intervals.

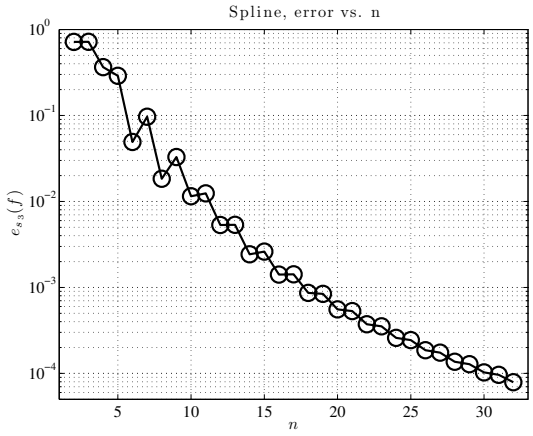

Figure 6: Errors $e_{s_3}(f)$ vs. n.

estimator $\widetilde{e}_1^H(f) := \frac{H^2}{8}\max_{x\in I}|f''(x)|$ such that $e_1^H(f) \leq \widetilde{e}_1^H(f)$. Since we assume that $H = \frac{b-a}{n}$, where n is the number of sub-intervals, we can compute the minimum number n_{min} that ensures that $e_1^H(f) < tol$ for some tolerance tol. We obtain n_{min} by imposing that $e_1^H(f) \leq \widetilde{e}_1^H(f) < tol$, i.e. by calculating the quantity $(b-a)\left(\frac{1}{8\,tol}\max_{x\in I}|f''(x)|\right)^{1/2}$. Since $f''(x) = \frac{2}{(1+x)^3}$, $a = 0$, $b = 5$, $\max_{x\in[0,5]}|f''(x)| = 2$, and $tol = 10^{-3}$, we obtain that $n_{min} = 80$.

b) Similarly to point a), we compute n_{min} as the quantity

$$(b-a)\left(\frac{C_0}{tol}\max_{x\in I}|f^{(4)}(x)|\right)^{1/4},$$

since the maximum error introduced in approximating $f(x)$ with the spline $s_3(x)$ corresponds to the choice $r = 0$ in the error bound. We obtain $n_{min} = 63$, since $f^{(4)}(x) = \frac{24}{(1+x)^5}$, $\max_{x\in[0,5]}|f^{(4)}(x)| = 24$, and $C_0 = 1$.

In order to ensure that the maximum error introduced in approximating $f'(x)$ with the derivative of the spline $s_3'(x)$ is inferior to the tolerance tol, we use the error bound with $r = 1$. We calculate n_{min} as the quantity $(b-a)\left(\frac{C_1}{tol}\max_{x\in I}|f^{(4)}(x)|\right)^{1/3}$. We obtain that $n_{min} = 145$.

c) The coefficients a_0 and a_1 of the least-squares approximating polynomial $\widetilde{f}_1(x) = a_0 + a_1 x$ are minimizers of the functional $\Phi(a_0, a_1)$. Such coefficients can be obtained by imposing the conditions: $\frac{\partial\Phi}{\partial a_0}(a_0, a_1) = 0$ and $\frac{\partial\Phi}{\partial a_1}(a_0, a_1) = 0$. We observe that:

$$\begin{aligned}\frac{\partial\Phi}{\partial a_0}(a_0,a_1) &= -2\sum_{i=0}^{2}[f(x_i)-(a_0+a_1x_i)] = -2\left(\sum_{i=0}^{2}f(x_i) - 3a_0 - a_1\sum_{i=0}^{2}x_i\right)\\ &= -2\left(\frac{13}{6} - 3a_0 - \frac{3}{2}a_1\right),\end{aligned}$$

$$\begin{aligned}\frac{\partial\Phi}{\partial a_1}(a_0,a_1) &= -2\sum_{i=0}^{2}x_i[f(x_i)-(a_0+a_1x_i)] = -2\left(\sum_{i=0}^{2}x_if(x_i) - a_0\sum_{i=0}^{2}x_i - a_1\sum_{i=0}^{2}x_i^2\right)\\ &= -2\left(\frac{5}{6} - \frac{3}{2}a_0 - \frac{5}{4}a_1\right),\end{aligned}$$

where we used the fact that $x_0 = 0$, $x_1 = \frac{1}{2}$, $x_2 = 1$, $f(x_0) = 1$, $f(x_1) = \frac{2}{3}$, and $f(x_2) = \frac{1}{2}$. Therefore, the coefficients a_0 and a_1 are obtained by solving the following linear system:

$$\begin{cases} -2\left(\frac{13}{6} - 3a_0 - \frac{3}{2}a_1\right) = 0 \\ -2\left(\frac{5}{6} - \frac{3}{2}a_0 - \frac{5}{4}a_1\right) = 0 \end{cases}, \quad \text{or} \quad \begin{cases} 3a_0 + \frac{3}{2}a_1 = \frac{13}{6} \\ \frac{3}{2}a_0 + \frac{5}{4}a_1 = \frac{5}{6} \end{cases}.$$

We obtain $a_0 = \frac{35}{36}$ and $a_1 = -\frac{1}{2}$, so that $\widetilde{f}_1(x) = \dfrac{35}{36} - \dfrac{1}{2}x$.

We observe that the previous linear system can be written in the compact form $A\,\mathbf{a} = \mathbf{g}$ by introducing a vector $\mathbf{a} \in \mathbb{R}^2$ containing the coefficients a_0 and a_1, the matrix $A \in \mathbb{R}^{2\times 2}$, and

the vector $\mathbf{g} \in \mathbb{R}^2$ such that:

$$A = \begin{bmatrix} 3 & \frac{3}{2} \\ \frac{3}{2} & \frac{5}{4} \end{bmatrix}, \qquad \mathbf{g} = \begin{bmatrix} \frac{13}{6} \\ \frac{5}{6} \end{bmatrix}.$$

We observe that $A = B^T B$ and $\mathbf{g} = B^T \mathbf{y}$, where the matrix $B \in \mathbb{R}^{3\times 2}$ and the vector $\mathbf{y} \in \mathbb{R}^3$ are defined as $B_{i,j} = (x_i)^j$ for $i = 0, 1, 2$ and $j = 0, 1$ and $\mathbf{y}_i = f(x_i)$ for $i = 0, 1, 2$, which read:

$$B = \begin{bmatrix} 1 & 0 \\ 1 & \frac{1}{2} \\ 1 & 1 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 1 \\ \frac{2}{3} \\ \frac{1}{2} \end{bmatrix}.$$

Solution IV (MATLAB)

a) Given a 2π-periodic function $f(x)$, $f : [0, 2\pi] \to \mathbb{R}$, and a set of $n+1$ equally spaced nodes $x_j = j\frac{2\pi}{n+1}$ for $j = 0, \ldots, n$, with $x_n < 2\pi$, the trigonometric interpolant for n *even* is defined as:

$$I_t f(x) = \sum_{k=-n/2}^{n/2} c_k \, e^{ikx},$$

for some coefficients $c_k \in \mathbb{C}$ with i the imaginary unit and $e^{ikx} = [\cos(k\,x) + i\,\sin(k\,x)]$. We remark that $I_t f(x_j) = f(x_j)$ for all $j = 0, \ldots, n$.

The MATLAB function `interpft` computes the $n+1$ unknown coefficients c_k by assuming that the interpolation nodes x_j are uniformly distributed in $[0, 2\pi)$ and provides the values of the trigonometric interpolant $I_t f(x)$ corresponding to a set of points in $[0, 2\pi)$ whose number is specified in input.

We are interested in constructing the trigonometric interpolant $I_t V(t)$ of the data points $\{(t_j, V_j)\}_{j=0}^{n}$, where the $n+1 = 11$ nodes $t_j = x_j \frac{T}{2\pi}$ are uniformly spaced in the period, as required by the MATLAB function `interpft`. For the purposes of graphical representation, we evaluate the trigonometric interpolant $I_t V(t)$ at $m\,(n+1)$ equally spaced points between $[0, 0.77)\,s$, with m "sufficiently" large, e.g. $m = 100$.

```
% Data couples
tj = [  0  0.07 0.14 0.21 0.28 0.35 0.42 0.49 0.56 0.63 0.70];
Vj = [ 194 184  177  156  142  160  168  166  170  178  187];
% Trigonometric interpolation
n = length( tj ) - 1; % n=10, n+1 = 11
m = 100;
t_values = linspace( 0, 0.77, m*(n+1) );
V_values = interpft( Vj, m*(n+1) );
plot( tj, Vj,'o', t_values, V_values, 'r--' );
% Ejection fraction
Ef = ( max( V_values ) - min( V_values ) ) / max( V_values )
%  Ef =
%       0.2714
```

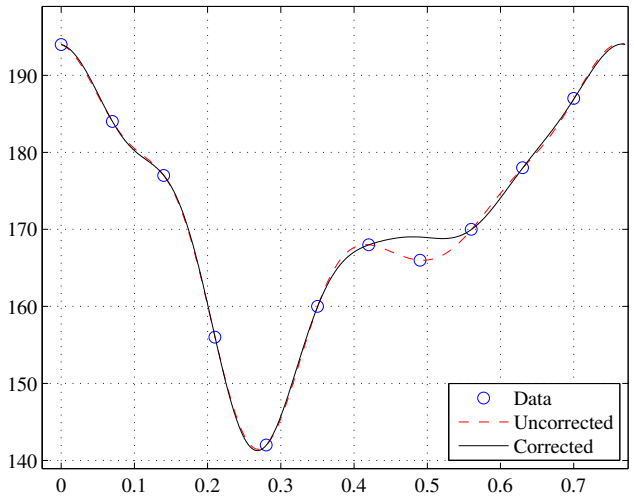

Figure 7: Trigonometric interpolants $I_t V(t)$ and $I_t \widetilde{V}(t)$ for $t \in [0, 0.77)\, s$.

The trigonometric interpolant $I_t V(t)$ is indeed periodic, as we can observe in Figure 7. The ejection fraction assumes the value $E_f = 0.2714$.

b) We use the following commands:

```
% Correction of data V_7
Vjc = Vj;
Vjc(7+1) = ( Vj(6+1) + Vj(8+1) ) / 2; %(Note: MATLAB index starts from 1)
% Trigonometric interpolation
Vc_values = interpft( Vjc, m*(n+1) );
hold on; plot( t_values, Vc_values, 'k-' );
legend('Data','Uncorrected','Corrected','Location','SouthEast')
% Ejection fraction (post-correction)
Efc = ( max( Vc_values ) - min( Vc_values ) ) / max( Vc_values )
%  Efc =
%       0.2720
```

It can be observed in Figure 7 that the modification at one interpolation node t_7 does not change the interpolant $I_t \widetilde{V}(t)$ much, as long as we stay away from the "bad" node, so that the corrected estimated ejection fraction $\widetilde{E}_f = 0.2720$ remains very similar to E_f.

Solution V (Theoretical)

Define the vector $\mathbf{r}(\mathbf{a}) = \mathbf{y} - B\mathbf{a} \in \mathbb{R}^n$. The least square solution $\tilde{\mathbf{a}} \in \mathbb{R}^m$ minimizes the quantity $\Phi(\mathbf{a})$ defined as

$$\Phi(\mathbf{a}) = \|\mathbf{r}(\mathbf{a})\|^2 = \|\mathbf{y} - B\mathbf{a}\|^2,$$

or alternatively as

$$\Phi(\mathbf{a}) = \sum_{i=1}^{n} r_i^2, \qquad r_i = y_i - \sum_{k=1}^{m} B_{ik} a_k, \quad i = 1, \ldots, n.$$

Since $\Phi(\mathbf{a})$ is a convex C^1 function, it is minimized when its gradient is zero. So, we proceed by computing its gradient:

$$\frac{\partial \Phi}{\partial a_j} = 2 \sum_{i=1}^{n} r_i \frac{\partial r_i}{\partial a_j}, \qquad j = 1, \ldots, m.$$

Since $\frac{\partial r_i}{\partial a_j} = -B_{ij}$, we can express the above equations as

$$\frac{\partial \Phi}{\partial a_j} = -2 \sum_{i=1}^{n} \Big(y_i - \sum_{k=1}^{m} B_{ik} a_k \Big) B_{ij}, \qquad j = 1, \ldots, m.$$

Finally, by imposing $\frac{\partial \Phi}{\partial \tilde{a}_j} = 0$, $j = 1, \ldots, m$, we have the relation

$$2 \sum_{i=1}^{n} \Big(y_i - \sum_{k=1}^{m} B_{ik} \tilde{a}_k \Big) B_{ij} = 0, \qquad j = 1, \ldots, m,$$

from which

$$\sum_{i=1}^{n} B_{ij} \sum_{k=1}^{m} B_{ik} \tilde{a}_k = \sum_{i=1}^{n} B_{ij} y_i, \qquad j = 1, \ldots, m.$$