and its three first time-derivatives are given by

$$\begin{aligned}\dot{\theta}(t) &= 7at^6 + 6bt^5 + 5ct^4 + 4dt^3 + 3et^2 + 2ft + g \\ \ddot{\theta}(t) &= 42at^5 + 30bt^4 + 20ct^3 + 12dt^2 + 6et + 2f \\ \theta^{(3)}(t) &= 210at^4 + 120bt^3 + 60ct^2 + 24dt + 6e\end{aligned}$$

From the conditions of eq.(93a) and the first condition of eq.(94), we have

$$e = f = g = 0, \qquad h = \theta_i^I$$

while the conditions of eq.(93b) and the second condition of eq.(94) yield four linear equations in $a$, $b$, $c$, and $d$, namely,

$$\begin{aligned}aT^3 + bT^2 + cT + d &= \frac{\theta_i^F - \theta_i^I}{T^4} \\ 7aT^3 + 6bT^2 + 5cT + 4d &= \frac{\dot{\theta}_i^F}{T^3} \\ 42aT^3 + 30bT^2 + 20cT + 12d &= 0 \\ 210aT^3 + 120bT^2 + 60cT + 24d &= 0\end{aligned}$$

Upon solving the four foregoing equations for their four unknowns, we obtain

$$\begin{aligned}a &= -\frac{20}{T^7}(\theta_i^F - \theta_i^I) + \frac{10}{T^6}\dot{\theta}_i^F \\ b &= \frac{70}{T^6}(\theta_i^F - \theta_i^I) - \frac{34}{T^5}\dot{\theta}_i^F \\ c &= -\frac{84}{T^5}(\theta_i^F - \theta_i^I) + \frac{39}{T^4}\dot{\theta}_i^F \\ d &= \frac{35}{T^4}(\theta_i^F - \theta_i^I) - \frac{15}{T^3}\dot{\theta}_i^F\end{aligned}$$

and hence, the seventh-degree polynomial sought is

$$\begin{aligned}\theta_i(t) = \theta_i^I &+ \left[-\frac{20}{T^7}(\theta_i^F - \theta_i^I) + \frac{10}{T^6}\dot{\theta}_i^F\right] t^7 + \left[\frac{70}{T^6}(\theta_i^F - \theta_i^I) - \frac{34}{T^5}\dot{\theta}_i^F\right] t^6 \\ &+ \left[-\frac{84}{T^5}(\theta_i^F - \theta_i^I) + \frac{39}{T^4}\dot{\theta}_i^F\right] t^5 + \left[\frac{35}{T^4}(\theta_i^F - \theta_i^I) - \frac{15}{T^3}\dot{\theta}_i^F\right] t^4\end{aligned}$$

**6.6** In order to have an idea of the magnitude of the error as the number of supporting points grows, we estimate the maximum error in each subinterval, as described presently. Furthermore, we can decompose the cycloidal function into a linear and a periodic part; we approximate only the latter, since the former does not contribute to the approximation error.

For a periodic cubic spline, similar to the solution of Exercise 6.1, we have the conditions, $s_1 = s_N$, $s_1^{'} = s_N^{'}$, $s_1^{''} = s_N^{''}$. The difference is that here we have $N$ intervals, $N+1$ points, and thus, matrices $\mathbf{A}$ and $\mathbf{C}$ are of $N \times N$, while

$$\mathbf{s} = [s_1, ..., s_N]^T, \qquad \mathbf{s}'' = [s_1'', ..., s_N'']^T. \tag{95}$$

Moreover,

$$\Delta x_k = \frac{1}{N}, \qquad k = 1, ..., N, \tag{96}$$

and for $i, j, k = 1, ..., N$,

$$\alpha_k = \frac{1}{N}, \qquad \alpha_{i,j} = \frac{2}{N}, \tag{97}$$

$$\beta_k = N, \qquad \beta_{i,j} = 2N. \tag{98}$$

With the above coefficients we can obtain the $\mathbf{A}$ and $\mathbf{C}$ matrices that, for periodic splines, are given in eq.(9.59a & b). Solving for $\mathbf{s}''$ in terms of $\mathbf{s}$ from eq.(6.58a), we obtain

$$\mathbf{s}'' = 6\mathbf{A}^{-1}\mathbf{C}\mathbf{s}. \tag{99}$$

With the foregoing calculations we have all the necessary information ($\mathbf{s}''$ and $\mathbf{s}$) to find the coefficients of the cubic polynomials appearing in eqs.(6.55a–d).

Now, the error at the ends of each subinterval vanishes. Hence, a plausible assumption is that the maximum absolute value of the error is attained at the midpoint of the interval. Note that a rigorous calculation of this maximum can be performed with a suitable optimization routine, that would use a *bisection* or *golden-section* subdivision of the interval.

A *Matlab* code, implemented as a function `error(N)` that returns the error $e_N$, for a given $N$, is included below:

```
function e = error(N)

% Definitions:
Delta = 1/N;                          % Subinterval length
D2 = Delta/2;                         % Half of Delta
ak = Delta;                           % alpha_k
bk = N;                               % beta_k
a2ij = 4*Delta;                       % 2*alpha_ij
bij = 2*bk;                           % beta_ij
pi2 = 2*pi;                           % 2*pi

t = ( 0 : Delta : 1);                 % subinterval step vector
for i = 1 : length(t) - 1,            % recursion from 1 to N

  % Cycloidal function
  cycloida(i) = t(i)- sin(pi2*t(i))/pi2;
  cycloidap(i) = 1 - cos(pi2*t(i));
  c(i) = cycloida(i) - t(i);          % c(i): the periodic part
  cp(i) = cycloidap(i) - 1;           % c'(i): the periodic part

  % Building matrices A and C
  if i == 1
    A(i,:) = [ a2ij, ak, zeros(1,N-3), ak ];
    C(i,:) = [ -bij, bk, zeros(1,N-3), bk ];
  else
    if i == N
      A(i,:) = [ ak, zeros(1,N-3), ak, a2ij ];
      C(i,:) = [ bk, zeros(1,N-3), bk, -bij ];
    else
      A(i,:) = [ zeros(1,i-2), ak, a2ij, ak, zeros(1,N-1-i) ];
      C(i,:) = [ zeros(1,i-2), bk, -bij, bk, zeros(1,N-1-i) ];
    end
  end
end
```

```
% Find s'' for the supporting points
s = c;                                % s(i) = c(i), i=1..N
sp = cp;                              % sp(i) = cp(i), i=1..N
spp = 6*inv(A)*C*sp';                 % finding s''

s(N+1) = s(1);                        % closing the cycle: s(N+1)
sp(N+1) = sp(1);                      % s'(N+1)
spp(N+1) = spp(1);                    % s''(N+1)

c(N+1) = c(1);                        % closing the cycle: c(N+1)

% Construct the spline function and calculate the error
for i = 1 : N,

  % Coefficients
  Ak(i) = ( spp(i+1) - spp(i) )/(6*Delta);
  Bk(i) = spp(i)/2;
  Ck(i) = ( s(i+1) - s(i) )/Delta - Delta*( spp(i+1) + 2*spp(i) )/6;
  Dk(i) = s(i);

  % Spline-function value at the midpoint of the ``i''-th subinterval
  sk = Ak(i)*(D2)^3 + Bk(i)*(D2)^2 + Ck(i)*(D2) + Dk(i);

  % Derivative value at the supporting points
  skp = Ck(i);
  skpp = 2*Bk(i);

  sc = [ sc, sk ];
  scp = [ scp, skp ];
  scpp = [ scpp, skpp ];

  % The periodic part of the cycloidal function at the midpoint
  cd =  - sin ( pi2*(t(i)+D2) )/pi2;

  % The derivative values at the supporting points are now
  cdp =  - cos ( pi2*(t(i)) );
  cdpp =  - pi2*sin ( pi2*(t(i)) );

  cc = [ cc, cd ];
  ccp = [ ccp, cd ];
  ccpp = [ ccpp, cd ];

end

% Error
ei = sc - cc;
eip = scp - ccp;
eipp = scpp - ccpp;

% Error norm
```

```
e = norm(ei, inf);
```

The results are summarized in Table 1. Note that, by increasing the number of subintervals by one

Table 1: Errors in the approximation of a cycloidal function with a periodic spline

| $N$ | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| $e_N$ | 0.0485 | 0.0124 | 0.0055 | 0.0031 | 0.0020 | 0.0014 | 0.0010 | 0.0008 | 0.0006 | 0.0005 |
| $e'_N$ | 1.0171 | 1.0126 | 1.0105 | 1.0100 | 1.0122 | 1.0126 | 1.0123 | 1.0119 | 1.0125 | 1.0126 |
| $e''_N$ | 36.947 | 38.855 | 39.206 | 39.328 | 39.384 | 39.414 | 39.432 | 39.444 | 39.452 | 39.457 |

order of magnitude, from 10 to 100, $e_N$ decreases by two orders of magnitude, from 0.0485 to 0.005. However, the errors in the first two derivatives remain virtually unchanged, with the error $e''_N$ exhibiting a slight increment as $N$ increases.

**6.8** We consider the 4-5-6-7 polynomial of eq.(6.27), namely,

$$P_{4567}(\tau) = -20\tau^7 + 70\tau^6 - 84\tau^5 + 35\tau^4. \tag{100}$$

The periodic part, $p_{4567}$, for $0 \leq \tau \leq 1$, is

$$p_{4567}(\tau) = -20\tau^7 + 70\tau^6 - 84\tau^5 + 35\tau^4 - \tau, \tag{101}$$

its first and second derivatives being:

$$p'_{4567}(\tau) = -140\tau^6 + 420\tau^5 - 420\tau^4 + 140\tau^3 - 1, \tag{102}$$

$$p''_{4567}(\tau) = -840\tau^5 + 2100\tau^4 - 1680\tau^5 + 420\tau^3. \tag{103}$$

Following the same procedure as with Exercise 6.6, we find the values of $p_{4567}(\tau)$ and $p'_{4567}(\tau)$. Then, we compute the $\mathbf{A}$ and $\mathbf{C}$ matrices, $s''(\tau_i)$, coefficients $A_k$, $B_k$, $C_k$, $D_k$, and, finally, the polynomial and the spline values for the midpoint of each subinterval, followed by the maximum error value.

A *Matlab* code, that returns the errors $e_N$, $e'_N$, $e''_N$, given $N$, is included below:

```
function [ e, ep, epp ] = p58(N)

% Definitions for computational efficiency
Delta = 1/N;                          % Subinterval length
D2 = Delta/2;                         % Half of Delta
ak = Delta;                           % alpha_k
bk = N;                               % beta_k
a2ij = 4*Delta;                       % 2*alpha_ij
bij = 2*bk;                           % beta_ij
pi2 = 2*pi;                           % 2*pi

t = ( 0 : Delta : 1 );                % subinterval endpoints
te = ( D2 : Delta : 1 - D2 );         % subinterval midpoints
for i = 1 : length(t) - 1,            % recursion from 1 to N

  % 4-5-6-7 Polynomial: the periodic part only
  po(i) = - 20*t(i)^7 + 70*t(i)^6 - 84*t(i)^5 + 35*t(i)^4 - t(i);
```

```
  pop(i) = - 140*t(i)^6 + 420*t(i)^5 - 420*t(i)^4 + 140*t(i)^3 -1;
  popp(i) = - 840*t(i)^5 + 2100*t(i)^4 - 1680*t(i)^3 + 420*t(i)^2;

  % Building matrices A and C
  if i == 1
    A(i,:) = [ a2ij, ak, zeros(1,N-3), ak ];
    C(i,:) = [ -bij, bk, zeros(1,N-3), bk ];
  else
    if i == N
      A(i,:) = [ ak, zeros(1,N-3), ak, a2ij ];
      C(i,:) = [ bk, zeros(1,N-3), bk, -bij ];
    else
      A(i,:) = [ zeros(1,i-2), ak, a2ij, ak, zeros(1,N-1-i) ];
      C(i,:) = [ zeros(1,i-2), bk, -bij, bk, zeros(1,N-1-i) ];
    end
  end
end

% Find s'' for the supporting points
s = po;                              % s(i) = c(i)
sp = pop;                            % s'(i) = c'(i)
spp = 6*inv(A)*C*sp';                % finding s''

s(N+1) = s(1);                       % closing the cycle: s
sp(N+1) = sp(1);                     % s'
spp(N+1) = spp(1);                   % s''

% Construct the spline function and calculate the error
for i = 1 : N,

  % Coefficients
  Ak(i) = ( spp(i+1) - spp(i) )/(6*Delta);
  Bk(i) = spp(i)/2;
  Ck(i) = ( s(i+1) - s(i) )/Delta - Delta*( spp(i+1) + 2*spp(i))/6;
  Dk(i) = s(i);

  % Spline function at midpoints of a subinterval
  sk = Ak(i)*(D2)^3 + Bk(i)*(D2)^2 + Ck(i)*(D2) + Dk(i);

  % Spline derivative values at the endpoints
  skp = Ck(i);
  skpp = 2*Bk(i);

  sc = [ sc, sk ];
  scp = [ scp, skp ];
  scpp = [ scpp, skpp ];

  % The periodic part of the 4-5-6-7 polynomial at the midpoints
  pom = - 20*te(i)^7 + 70*te(i)^6 - 84*te(i)^5 + 35*te(i)^4 - te(i);

  % The periodic part of the 4-5-6-7 polynomial at the endpoints
```

```
  pomp = - 140*t(i)^6 + 420*t(i)^5 - 420*t(i)^4 + 140*t(i)^3 - 1;
  pompp = - 840*t(i)^5 + 2100*t(i)^4 - 1680*t(i)^3 + 420*t(i)^2;

  poc = [ poc, pom ];
  pocp = [ pocp, pomp ];
  pocpp = [ pocpp, pompp ];

end;
```

Note that the approximation error can be estimated using the *Matlab* spline routines. Below we include the *Matlab* code that estimates errors and numbers of subintervals required to attain the desired accuracy:

```
function [eN,Ne,epN,Nep,eppN,Nepp] = p58(epsilon)

N = 4; % initial number of subintervals for the iterations
dN = 1; % initial increment for N
e = 2, ep = 1, epp = 1; % initial value for the error

disp('Function error');
while (e > epsilon),             % while iteration
  N = N + dN; disp(N), disp(e),  % increasing the number of subintervals
  Delta = 1/N;                         % subinterval length
  Delta2 = Delta/2;                           % half of Delta
  te = ( 0 : Delta : 1 );                % subinterval endpoints: N + 1
  tm = ( Delta2 : Delta : 1 - Delta2 );     % subinterval midpoints: N
  % 4-5-6-7 Polynomial minus the linear part
  pm = - 20*tm.^7 + 70*tm.^6 - 84*tm.^5 + 35*tm.^4 - tm; % at midpoints
  pe = - 20*te.^7 + 70*te.^6 - 84*te.^5 + 35*te.^4 - te; % at endpoints
  sm = ppval(csape(te,pe,[0 0]),tm); % periodic cubic spline at midpoints
  ev =  sm - pm;   % function error at midpoints
  e = norm(ev, inf);      % the maximum midpoint error over all subintervals
end % while

% Documentation
Ne = N, eN = e, % return of N and e
figure(1)
subplot(3,1,1), plot(ev)
dN = dN*5,  % increase in the search order for faster search

disp('First derivative error');
while (ep > epsilon),                % while iteration
  N = N + dN; disp(N), disp(ep),  % increasing the number of subintervals
  Delta = 1/N;                           % subinterval length
  Delta2 = Delta/2;                             % half of Delta
  te = ( 0 : Delta : 1 );                % subinterval endpoints: N + 1
  tm = ( Delta2 : Delta : 1 - Delta2 );        % subinterval midpoints: N
  % First Derivative of 4-5-6-7 Polynomial minus the linear part
  pep = - 140*te.^6 + 420*te.^5 - 420*te.^4 + 140*te.^3 -1; % at endpoints
  % 4-5-6-7 Polynomial minus the linear part
```

```
  pe = - 20*te.^7 + 70*te.^6 - 84*te.^5 + 35*te.^4 - te; % at endpoints
  sep = ppval(fnder(csape(te,pe,[0 0]),1),te); % p.c.s.' at endpoints
  epv = sep - pep;  % first derivative error at endpoints
  ep = norm(epv, inf);    % the maximum midpoint error over all subintervals
end % while

% Documentation
Nep = N, epN = ep, % recording N and ep
subplot(3,1,2), plot(epv)
dN = dN*10,  % increase in the search order for faster search

disp('Second derivative error');
while (epp > epsilon),               % while iteration
  N = N + dN; disp(N),disp(epp),   % increasing the number of subintervals
  Delta = 1/N;                          % subinterval length
  Delta2 = Delta/2;                           % half of Delta
  te = ( 0 : Delta : 1 );               % subinterval endpoints: N + 1
  tm = ( Delta2 : Delta : 1 - Delta2 );        % subinterval midpoints: N
  % First Derivative of 4-5-6-7 Polynomial minus the linear part
  pepp = - 840*te.^5 + 2100*te.^4 - 1680*te.^3 + 420*te.^2;
  % 4-5-6-7 Polynomial minus the linear part
  pe = - 20*te.^7 + 70*te.^6 - 84*te.^5 + 35*te.^4 - te; % at endpoints
  sepp = ppval(fnder(csape(te,pe,[0 0]),2),te); % p.c.s.'' at midpoints
  eppv =  sepp - pepp;  % second derivative error at midpoints
  epp = norm(eppv, inf);   % the maximum midpoint error over all subintervals
end % while

% Documentation
Nepp = N, eppN = epp, % recording N and epp
subplot(3,1,3), plot(eppv)
```

The results obtained for an error less than 0.0001 are given in Table 2.

Table 2: Error values

| | | | |
|---|---|---|---|
| $N$ | 15 | $e_N$ | 8.7347e-05 |
| $N$ | 60 | $e'_N$ | 8.9181e-05 |
| $N$ | 960 | $e''_N$ | 9.5397e-05 |

# 7 Dynamics of Serial Robotic Manipulators

**7.3** In order to apply Algorithms 7.4.1 and 7.4.2 to obtain the necessary torque at each joint to perform the desired maneuver, we must first obtain the joint rates and accelerations. Since the velocity and acceleration of the wrist center $C$ are given, as well as the angular velocity and acceleration of the end-effector, the joint rates and accelerations are readily obtained using the Jacobian matrix. For a decoupled manipulator, we have

$$\mathbf{J}_{11}\dot{\boldsymbol{\theta}}_a + \mathbf{J}_{12}\dot{\boldsymbol{\theta}}_w = \boldsymbol{\omega} \tag{104a}$$

$$\mathbf{J}_{21}\dot{\boldsymbol{\theta}}_a = \dot{\mathbf{c}} \tag{104b}$$