2.11. MATLAB‘o programų konstravimas ir testavimas

Funkcijų konstravimas. Programų rašymas yra kūrybinis procesas ir jo pilnai formalizuoti neįmanoma. Tačiau programų kūrimas bus aiškesnis ir jų rašymo bei testavimo laikas bus trumpesnis, jei laikysimės praktikoje patikrintų rekomendacijų.

Pagrindinis MATLAB‘o programų elementas yra funkcijos. Apie MATLAB‘o funkcijų sintaksę kalbėjome anksčiau. Čia aptarsime jų struktūrą.

Rekomenduotiną bazinę MATLAB‘o funkcijų struktūrą sudaro tokios dalys.

1. ***Funkcijos antraštė.***
2. ***Įžanga – prologas.***
3. ***Nebūtinų įėjimo parametrų reikšmių priskirimas ir būtinų parametrų reikšmių tikrinimas.***
4. ***Funkcijos kūnas.***
5. ***Nebūtinų išėjimo parametrų reikšmių priskirimas.***

Funkcijos ***antraštė*** yra būtina dalis, o ***prologas*** – rekomenduotina dalis. Funkcijos antraštės sintaksė buvo aptarta 2.8 paragrafe.

Prologą sudaro komentarų eilutės. Kaip buvo minėta anksčiau, pirmoji prologo eilutė vadinama ***H1*** eilute ir ji yra komentaras, prasidedantis funkcijos vardu. Kad funkcijos vardas išsiskirtų tekste, patartina jį rašyti didžiosiomis raidėmis.

Prologe rekomenduotina nurodyti funkcijos paskirtį, paaiškinti įėjimo ir išėjimo parametrus, nurodyti literatūrą, kurioje aprašytas funkcijos realizuojamas metodas ir pan. Prologo eilučių skaičius kartais gali būti didesnis už funkcijos komandinių eilučių skaičių.

Kaip buvo minėta 2.8 paragrafe, prologo tekstas yra atspausdinamas komandiniame lange vykdant komandą:

help *funkcijos vardas.*

Prologas baigiamas tuščia eilute. Komentarai po šios eilutės – nespausdinami.

Dažnai MATLAB‘o funkcijos sudaromos taip, kad, kreipiantis į jas, galima nurodyti ne visus ***įėjimo parametrus***, o tik būtiną jų poaibį. Todėl, prieš atliekant pagrindinius skaičiavimus, reikia apibrėžti nebūtinus (likusius) įėjimo parametrus, o taip pat patikrinti perduodamų parametrų reikšmes. Tam skirta trečioji funkcijos struktūros dalis.

Analogiška padėtis yra ir su išėjimo parametrais: kreipiantis į procedūrą, nebūtina nurodyti visus jos išėjimo parametrus, o tik būtinus. Tam skirta baigiamoji funkcijos dalis, kurioje priskiriamos nebūtinų išėjimo parametrų reikšmės.

Funkcijos įėjimo ir išėjimo parametrų skaičių nustato MATLAB‘o funkcijos: ***nargin*** ir ***nargout***.

Pastaba. *Kreiptis į MATLAB‘o funkcijas galima su mažesniu įėjimo ir išėjimo parametrų skaičiumi. Tačiau MATLAB‘as nurodys klaidą, jei šių parametrų skaičius bus didesnis nei maksimaliai galimas parametrų skaičius.*

Aišku, kad konstruojant funkciją pagrindinis dėmesys skiriamas ***funkcijos kūnui*** – ***pagrindiniams skaičiavimams***.

Rašant pagrindinių skaičiavimo tekstą, patartina:

- Stengtis, kad funkcijos tekstas nebūtų labai ilgas ir tilptų viename lape;
- Funkcijos tekstas turėtų būti struktūrizuotas: operatoriai turėtų būti atliekami nuosekliai iš viršaus žemyn;
- Aišku, kad funkcijos struktūra bus aiškesnė, jei, rašydami sąlyginio valdymo komandą ***if...end*** bei ***for*** arba ***while*** ciklus, naudosime įtraukas;

  Pvz.
```
if sign(pa)*sign(pc) > 0
        a=c;
    elseif abs(pc)<=eps
            p=false;
            s=c;
        else
            b=c;
    end
```
- Naudoti prasmingus kintamųjų vardus. MATLAB‘o kintamųjų vardų sudarymo taisyklės buvo aptartos 2.2 paragrafe. Aišku, kad nereikia vartoti ir per daug ilgų vardų, nes tai apsunkina programos skaitymą ir, juos rašant, galima lengvai įvelti klaidą.
- Programos tekstas bus aiškesnis, jei rašysime ***komentarus.*** Komentarai yra būtini, jei sudaryta funkcija bus išsaugota ir naudojama skaičiavimuose. Pirmiausia komentarai turi paaiškinti įėjimo ir išėjimo parametrų prasmę. Be to, jie turi paaiškinti atskirų funkcijos dalių turinį, o taip pat atskleisti programos kodo loginius ir matematinius ryšius.

  Komentarų rašymas vadinamas ***programų dokumentavimu***. Literatūroje yra pateiktkiamos dokumentavimo rekomendacijos, kurias apibendrintai galima išvardinti taip:
  1. Jei programa yra klaidinga, tai komentarų prasmė nėra svarbi;
  2. Nėra didelė blogybė, jei teisingo kodo komentarai nėra išsamūs;
  3. Kodas turi pats dokumentuoti save. Kitaip tariant, kodas turi būti parašytas taip, kad jis būtų nesunkiai suprantamas su minimaliu komentarų skaičiumi;
  4. Komentarai turi suteikti papildomą informaciją, o ne kaip papūga kartoti programos kodą;
  5. Prasmingi kintamųjų vardai ir komentarai bei kodo išdėstymas, pabrėžiantis loginę programos struktūrą, yra geros programos požymiai.

Programos sudarymo etapai **.** Paprastus skaičiavimus MATLAB‘e galima atlikti interaktyviai, užrašius keletą reikalingų komandų. Aišku, kad tai priklauso nuo profesionalumo: kaip gerai skaičiuotojas susipažinęs su MATLAB‘u, jo galimybėmis, o taip pat su skaitiniais metodais.

Sudėtingų skaičiavimų programos kodo sudarymas, paprastai susideda iš keturių etapų.

1. Prieš rašant programos kodą, sudaromas bendras sprendimo planas. Planą lengviau sudaryti keliais lygiais, pradedant stambiais žingsniais, o po to juos smulkinant, kol jų realizacija tampa visiškai aiški.
2. Antrame etape, remiantis pirmo etapo planu, žodžiais aprašomi atskirų programų moduliai, kurie spręstų vieną ar kelis gimininguss uždavinius. Pirmi du etapai sudaro skaitinio sprendinio projektavimo fazę.
3. Trečiame etape, remiantis modulių aprašais, sudaromos tos modulius realizuojančios MATLAB'o funkcijos (jų kodas). Šio etapo darbas yra lengvesnis, jei pirmi du etapai yra kruopščiai atlikti.
4. Ketvirtas etapas yra skirtas derinti ir testuoti programą. Prieš atliekant skaičiavimus, turime būti įsitikinę, kad programa veikia teisingai.

Programos derinimas ir testavimas. Programa gali blogai veikti dėl įvairių priežasčių. Kartais genda aparatūra, įėjimo duomenyse yra klaidingų simbolių arba komandose yra sintaksinių ar loginių klaidų. Programinės įrangos klaida angliškai vadinama „***bug***" (vabalas), o klaidų taisymas angliškai vadinamas „***debugging***".

Terminą „bug" sutrikusiai kompiuterio veiklai aprašyti pirmoji panaudojo kompiuterio MARK II programuotoja Grace Murray Hopper. MARK II buvo vienas iš pirmųjų kompiuterių, sukurtų dar prieš tranzistoriaus išradimą. Kompiuteris MARK II buvo sukonstruotas iš vakuuminių lempų ir elektromagnetinių relių. Kartą, sugedus MARK II, G. Hopper nustatė, kad gedimą sukėlė naktinis drugys patekęs tarp elektromagnetinės relės kontaktų. Žuvęs drugys sukėlė trumpą jungimą elektros grandinėje, ko pasėkoje kompiuterio veikla sutriko. G. Hopper išėmė drugį, įdėjo jį į laboratorijos žurnalą ir jame parašė, kad gedimo priežastis buvo vabalas (bug) kompiuteryje. Todėl dabar klaidų ieškojimas programose vadinamas „vabalų šalinimu", o programos, kurios padeda rasti programose klaidas, angliškai vadinamos „***debugger***".

Programos derinimą siūlome pradėti nuo atskirų žemiausio lygio funkcijų derinimo. Toliau, suderintos funkcijos, jungiamos į aukštesnio lygio modulius ir t.t. Toks programos derinimo metodas vadinamas ***„derinimas iš apačios į viršų"***.

Kitas alternatyvus programų derinimo būdas yra ***„iš viršaus į apačią".*** Šiuo atveju pirmiausia derinamas aukščiausio lygio modulis, kuris kreipiasi į fiktyvias žemesnio lygio funkcijas. Fiktyvi funkcija turi tik funkcijos antraštė, o jos kūnas, pavyzdžiui, yra komanda, spausdinanti kokį tai tekstą arba įėjimo parametrų reikšmes. Tuo būdu, pirmiausia suderinama visos programos loginė struktūra, o po to derinami žemesnio lygio programos moduliai ir t.t.

MATLAB'as turi patogias programos derinimo priemones, kurios įgalina programos vykdymą stabdyti ties norimais operatoriais. Taip pat jos įgalina programą vykdyti po vieną komandą. Be to, sustabdžius programos vykdymą, galime paprastai sužinoti tarpinių skaičiavimų rezultatus. Tam tikslui komandų lange reikia parašyti tų kintamųjų vardus.

Norint, kad skaičiuojant būtų spausdinami tarpiniai rezultatai, tereikia panaikinti komandos, apskaičiuojančios mus dominantį rezultatą, gale esantį kabliataškį arba programoje parašyti rezultato vardą.

Aišku, kad 2.5 paragrafe aprašytos komandos: ***pause, keyboard, break, return*** *ir* ***try-catch*** taip pat yra skirtos programų derinimui ir, tinkamai naudojamos, palengvina programos derinimą.

Programos derinimas palengvės, jei, sudarant funkcijas, jose bus realizuotos apsauginės priemonės. Pavyzdžiui, funkcijos pradžioje patartina tikrinti įėjimo parametrų reikšmes: ar jos neišeina už leistinų ribų. Paprastas pavyzdys, jei funkcija skaičiuoja *log(x)*, tai būtina patikrinti, ar *x* reikšmė yra teigiama. Taip pat į programos kodą patartina įtraukti komandas, pvz., ***lasterr*** (žr. 2.5 paragrafą), atspausdinančias pranešimus apie klaidos pobūdį ar vietą. Tokie pranešimai palengvina programos derinimą.

Programa paprastai derinama sprendžiant uždavinį, kuriam tarpiniai ir galutiniai skaičiavimo rezultatai yra žinomi. Tai, pavyzdžiui, gali būti rankiniu būdu išspręstas uždavinys arba pradiniai duomenys parinkti taip, kad sprendinys iš anksto yra žinomas.

Tarkime, kad norime sukonstruoti kubinį interpoliacinį splainą $y = g(x)$. Tada, jei paimtume bet kokį kubinį polinomą $y = f(x)$, sudarytume to polinomo reikšmių lentelę $(x_i, y_i), i = \overline{0, N}$, o kraštines sąlygas parinktume taip, kad, pavyzdžiui, $g''(x_0) = f''(x_0), g''(x_N) = f''(x_N)$, tai apskaičiuotas interpoliacinis splainas $y = g(x)$ ir polinomas $y = f(x)$ sutampa.

Pavyzdys. Sudarykime programą, kuri apskaičiuotų interpoliacinį kubinį splainą, kai žinomi interpoliavimo mazgai $(x_i, y_i), i = \overline{0, N}$.

Tam, kad šį uždavinį išspręstume pirmiausia ***turime prisiminti splaino sąvoką, jo užrašymo formas ir interpoliavimo uždavinio formulavimą.***

Splainai tai funkcijos, sudarytos iš polinomų dalių. Jei plokštumoje pažymėsime keletą taškų ir per juos "išraitysime" standžią plieninę liniuotę, tai liniuotė įgaus kubinio splaino formą. Kubinis splainas yra sudarytas iš kubinio polinomo dalių. Liniuotės forma tarp gretimų taškų, t.y. kiekviename intervale $[x_{i-1}, x_i], i = \overline{1, N}$ bus kubinis polinomas. Aišku, kad kubiniai polinomai tarp skirtingų gretimų taškų porų bus skirtingi, tačiau polinomų susidūrimo taškuose tų polinomų ir jų išvestinių iki antros eilės imtinai reikšmės iš kairės ir dešinės bus lygios. Kalbant griežtai, turėsime ***kubinį defekto vienas splainą***. Defektas parodo kiek aukščiausios eilės išvestinių susidūrimo taškuose iš kairės ir dešinės tarpusavyje nėra lygios.

Yra įvairių splaino užrašymo formų. Apie jas kalbėsime vėliau. Kubiniam splainui užrašyti pasirinkime dalimis polinominę formą.

Tada splaininės interpoliacijos uždavinys formuluojamas taip.

Uždavinio formulavimas. Duota funkcijos $y = f(x)$ reikšmių lentelė (x_i, y_i), $i = \overline{0, N}$. Reikia rasti kubinį splainą $y = g(x)$, tenkinantį Lagranžo interpoliavimo sąlygą $g(x_i) = y_i, i = \overline{0, N}$.

Ar šis uždavinys apibrėžtas?

Kaip matyti iš apibrėžimo, kubinį splainą apibūdina $4N$ koeficientai: kiekviename intervale $[x_{i-1}; x_i]$ kubinį polinomą nusako keturi koeficientai, o intervalų skaičius lygus N.

Kiek turime lygčių, siejančių tuos koeficientus?

Turime $N+1$ interpoliavimo sąlygą. Kiekviename vidiniame intervalo taške polinomai turi tenkinti tris sąlygas, vadinasi, turime dar $3(N-1)$ lygtis. Taigi $4N$ kubinio splaino koeficientai privalo tenkinti $4N-2$ lygtis. Kad interpoliacinis kubinis splainas būtų apibrėžtas vienareikšmiškai, reikia pasirinkti dvi papildomas sąlygas, vadinamas kraštinėmis sąlygomis. Dažniausiai pasirenkamos šios kraštinės sąlygos:

1) $g''(x_0)=g''(x_N)=0$ (jos vadinamos natūraliomis kraštinėmis sąlygomis ir rodo, kad plieninės liniuotės, išraitytos per interpoliavimo taškus, galai paliekami laisvai kaboti);

2) $g''(x_0)=f''(x_0),\quad g''(x_N)=f''(x_N)$;

3) $g'(x_0)=f'(x_0),\quad g'(x_N)=f'(x_N)$;

4) $g'(x_0)=g'(x_N),\quad g''(x_0)=g''(x_N)$ (jos vadinamos periodinėmis kraštinėmis sąlygomis);

5) $g'''(x_1-0)=g'''(x_1+0),\quad g'''(x_{N-1}-0)=g'''(x_{N-1}+0)$ (šios sąlygos reiškia, kad intervaluose $[x_0;x_1]$ ir $[x_1;x_2]$, taip pat $[x_{N-2};x_{N-1}]$ ir $[x_{N-1};x_N]$ atitinkamai yra ta pati kubinio polinomo išraiška).

Interpoliaciniam splainui konstruoti pasirinkime antrąsias kraštines sąlygas:

$g''(x_0)=f''(x_0),\quad g''(x_N)=f''(x_N)$.

Jei $f''(x_0)=f''(x_N)=0$, tai šios sąlygos sutaps su natūraliomis kraštinėmis sąlygomis.

Literatūroje yra pateikta iterpoliacinio kubinio splaino dalimis polinominė forma: ***kubinio splaino išraiška intervale*** $[x_{i-1};x_i]$ ***yra tokia***:

$$g(x)=m_{i-1}\frac{(x_i-x)^3}{6h_i}+m_i\frac{(x-x_{i-1})^3}{6h_i}+\left(y_{i-1}-\frac{m_{i-1}h_i^2}{6}\right)\frac{x_i-x}{h_i}+\left(y_i-\frac{m_ih_i^2}{6}\right)\frac{x-x_{i-1}}{h_i}.$$

Šioje formulėje:

- $m_i=g''(x_i), i=\overline{0,N}$ - splaino antrųjų išvestinų reikšmės interpoliavimo mazguose,
- $h_i=x_i-x_{i-1}, i=\overline{1,N}$ - i – ojo intervalo ilgis.

Tam, kad kubinis splainas, tenkinantis antrąsias kraštines sąlygas, būtų interpoliacinis, splaino antrųjų išvestinių $m_i, i=\overline{1,N-1}$ reikšmės turi būti tiesinės lygčių sistemos $Am=Hy$ sprendinys; čia A — simetrinė kvadratinė $(N-1)$-osios eilės triįstrižainė matrica

$$A=\begin{pmatrix}
\frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & \dots & 0 & 0\\
\frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & \dots & 0 & 0\\
0 & \frac{h_3}{6} & \frac{h_3+h_4}{3} & \dots & 0 & 0\\
\dots & \dots & \dots & \dots & \dots & \dots\\
0 & 0 & 0 & \dots & \frac{h_{N-1}}{6} & \frac{h_{N-1}+h_N}{3}
\end{pmatrix},$$

H — stačiakampė $(N-1)\times(N+1)$ matrica

$$H=\begin{pmatrix} \frac{1}{h_1} & -\left(\frac{1}{h_1}+\frac{1}{h_2}\right) & \frac{1}{h_2} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{h_2} & -\left(\frac{1}{h_2}+\frac{1}{h_3}\right) & \frac{1}{h_3} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{h_{N-1}} & -\left(\frac{1}{h_{N-1}}+\frac{1}{h_N}\right) & \frac{1}{h_N} \end{pmatrix},$$

m ir y — matricos stulpeliai

$$m=\begin{pmatrix} m_1 \\ m_2 \\ \dots \\ m_{N-1} \end{pmatrix},\ y=\begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_N \end{pmatrix}.$$

Be to, jei pažymėsime $d=Hy$, tai d_1 ir d_{N-1} yra perskaičiuojami pagal formules:

$d_1=d_1-m_0*h_1/6$ ir $d_{N-1}=d_{N-1}-m_N*h_N/6$.

Sistema $Am=Hy$ yra tiesinių lygčių sistema su triįstrižaine matrica A, kurios pagrindinės įstrižainės elementai didesni už kitų eilutės elementų sumą. Taigi ši sistema yra suderintoji ir turi vienintelį sprendinį, kurį patogu apskaičiuoti perkelties metodu.

Vadinasi, norint sukonstruoti interpoliacinį kubinį splainą, reikia suformuoti ir vieną kartą išspręsti $Am=Hy$ lygčių sistemą.

Kubinį splainą apibūdina trys masyvai:

$\left.\begin{matrix} x[0..N] \\ y[0..N] \end{matrix}\right\}$ — duotieji interpoliavimo taškai,

$m[0..N]$ — lygčių sistemos $g(x)$ sprendinys, t. y. apskaičiuotos splaino antrųjų išvestinių reikšmės.

Norint apskaičiuoti funkcijos $y=f(x)$ reikšmę pagal kubinį interpoliacinį splainą, reikia:

1) rasti intervalą $[x_{i-1};x_i]$, kuriam priklauso argumento reikšmė x, t. y. $x\in[x_{i-1};x_i]$;

2) pagal $g(x)$ formulę apskaičiuoti splaino reikšmę taške x.

Iš pateikto teorinio nagrinėjimo matome, kad tikslinga sukonstruoti funkciją, kuriai pateikus interpoliavimo taškus, būtų suformuota ir išspręsta tiesinių lygčių $Am=Hy$ sistema, kurios sprendinys bus splaino antrųjų išvestinių reikšmės. Kadangi ši sistema yra triįstrižaininė su dominuojančia pagrindine įstrižaine, tai ją spręsime perkelties metodu.

Perkelties metodas sistemai $a_ix_{i-1}-b_ix_i+c_ix_{i+1}=d_i$, čia $i=\overline{1,n}$, $a_1=c_n=0$, aprašomas formulėmis:

Tiesioginis etapas

$s_{i+1}=c_i/(b_i-a_is_i)$,

$e_{i+1}=(a_ie_i-d_i)/(b_i-a_is_i)$, $i=\overline{1,n}$, $s_1=e_1=0$, nes $a_1=0$.

Atvirkštinis etapas

$x_i = s_{i+1}x_{i+1} + e_{i+1}, \quad i = \overline{n, 1}, \quad x_{n+1} = 0,$ nes $c_n = 0$.

Aišku, kad, norint apskaičiuoti splaino reikšmes, reikia sudaryti funkciją, kuriai pateikus interpoliavimo taškus (x_i, y_i), bei argumento $t \in [x_0, x_N]$ reikšmes, būtų apskaičiuojamos splaino reikšmės.

Funkciją, kuri apskaičiuoja splaino antrųjų išvestinių reikšmes, pavadinkime ***splin***, perkelties metodą realizuojančią funkciją pavadinkime ***perkelt***, o funkciją, apskaičiuojančia splaino reikšmes, - ***splg***. Rašydami šių funkcijų tekstus, stengsimės skaičiavimus vektorizuoti (žr. 2.5 paragrafą).

Žemiau pateikti šių funkcijų tekstai.

```
function fm=splin(x,y,m0,mn)
% SPLIN funkcija apskaičiuoja kubinį interpoliacinį splainą,
% kai žinomos jo antrųjų išvestiniu reikšmės kraštiniuose taškuose.
% Įėjimo parametrai
% x   - interpoliavimo mazgu masyvas,
% y   - interpoliuojamos funkcijos reikšmės mazguose,
% m0  - splaino antrosios išvestines reikšmė pirmajame mazge,
% mn  - splaino antrosios išvestines reikšme paskutiniame mazge.
% Išėjimo parametrai
% fm  - splaino antrųjų išvestinių reikšmės interoliavimo mazguose.

n=numel(x); % n - interpoliavimo mazgu skaičius
% Atminties išskirimas a, b, c, d, ir fm masyvams
a=zeros(1,n-1); b=zeros(1,n-1); c=zeros(1,n-1); d=zeros(1,n-1);
fm=zeros(1,n-1);
% Lygčių sistemos Am=Hy formavimas
h=x(2:n)-x(1:n-1);
a=h(2:n-2)/6;
a=[0 a];
b=-(h(1:n-2)+h(2:n-1))./3;
c=[a(2:n-2) 0];
p=ones(1,n-2);
d(1:n-2)=y(1:n-2)./h(1:n-2)-y(2:n-1).*(p./h(1:n-2)+...
p./h(2:n-1))+y(3:n)./h(2:n-1);
 d(1)=d(1)-m0*h(1)/6;
 d(n-2)=d(n-2)-mn*h(n-1)/6;
% Sprendžiama lygčių sistema Am=Hy
fm=perkelt(a,b,c,d);
% Formuojamas antrųjų išvestinių reikšmių masyvas fm
fm=[m0 fm mn];

function x=perkelt(a,b,c,d)
% PERKELT sprendžia triįstrižainine tiesinių lygčių sistemą
% perkelties metodu
% Įėjimo parametrai
% a, b ir c - vienmačiai masyvai, nusakantys sistemos kairiąją pusę,
% d         - sistemos laisvųjų narių masyvas.
% Išėjimo parametrai
% x         - sistemos sprendinys.
% Tiesioginis etapas
n=numel(b);
```

```
s(1)=0; e(1)=0;
for i=1:n
    t=b(i)-a(i)*s(i);
    s(i+1)=c(i)/t;
    e(i+1)=(a(i)*e(i)-d(i))/t;
end
% Atvirkštinis etapas
x(n)=e(n+1);
for i=n-1:-1:1
    x(i)=s(i+1)*x(i+1)+e(i+1);
end

function gt=splg(x,y,fm,t);
%  SPLG apskaičiuoja kubinio interpoliacinio splaino
% reikšmes, kai splinas nusakytas dalimis polinomine forma.
% Įėjimo parametrai
% x - tinklelio mazgai,
% y - funkcijos reikšmes,
% fm - splaino antrųjų išvestinių reikšmės,
% t - masyvas argumento reikšmių, kuriuose norime apskaičiuoti
% splaino reikšmes,
% Išėjimo parametrai
% gt - splaino reikšmės taškuose x=t.

m=numel(t); % argumento reikšmių skaičius
n=numel(x); % interpoliavimo mazgų skaičius
% Tikriname, ar yra argumento t reikšmių už intervalo
%  (x(1),x(n)) ribų
if (t(1)<x(1)) | (t(m)>x(n))
    disp('yra argumentų t už intervalo ribų')
    gt=nan;
    return
end
h=x(2:n)-x(1:n-1); % intervalų tarp interpoliavimo mazgų masyvas
tm=repmat(t',1,n);
xx=repmat(x,m,1);
% Apskaičiuojame intervalus, kuriems priklauso argumentų t reikšmės
ind=n-sum((xx>=tm)');
if ind(1)==0
    ind(1)=1;
end
for i=1:m
    xi1(i)=x(ind(i));  xi(i)=x(ind(i)+1);
    yi1(i)=y(ind(i));  yi(i)=y(ind(i)+1);
    hi(i)=h(ind(i));
    fmi1(i)=fm(ind(i));
    fmi(i)=fm(ind(i)+1);
end
% Apskaičiuojame splaino reikšmes taškuose x=t
gt=fmi1.*(xi-t).^3./(6*hi)+fmi.*(t-xi1).^3./(6*hi)+...
    (yi1-fmi1.*hi.^2./6).*(xi-t)./hi+...
    (yi-fmi.*hi.^2./6).*(t-xi1)./hi;
```

Funkciją ***perkelt*** derinsime spręsdami trijįstrižaininę sistemą, kurios sprendinį žinome. Tam tikslui sudarykime masyvus: *a, b, c* ir *xt*, apskaičiuokime laisvųjų narių masyvą *d* laikydami, kad sprendinys yra *xt*, ir kreipkimės į funkciją ***perkelt.*** Funkcija

perkelt grąžins sprendinį *x*, kuris turi sutapti su masyvu *xt.* Kitaip tariant, spręsime sistemą:

$a_i xt_{i-1} - b_i xt_i + c_i xt_{i+1} = d_i, i = \overline{1,n}$.

Vadinasi, jei komandų lauke, pavyzdžiui, surinktume komandas:

```
>> a=[0 1 -2 4 2];
>> b=[5 -8 7 10 12];
>> c=[1 -3 2 -2 0];
>> xt=[1 1 1 1 1];
>> d=a.*xt-b.*xt+c.*xt
d = -4     6    -7    -8   -10
>> x=perkelt(a,b,c,d),
```

tai, kaip ir tikėjomės, turėsime sprendinį:

```
x = 1.0000    1.0000    1.0000    1.0000    1.0000.
```

Funkcijas ***splin*** ir ***splg*** derinkime kubiniu splainu interpoliuodami kubinį polinomą, pavyzdžiui, $y = x^3 - 4x^2 + 1$. Jei splaino antrųjų išvestinių reikšmes kraštiniuose interpoliavimo mazguose prilyginsime kubinio polinomo antrųjų išvestinių reikšmėms tuose pačiuose mazguose, tai interpoliacinis splainas ir kubinis polinomas privalės sutapti. Tai reiškia, kad funkcija ***splin*** grąžins fm_i reikšmes, kurios bus lygios $6x_i - 8$ visuose interpoliavimo mazguose x_i.

Jei splaino antrųjų išvestinių reikšmės interpoliavimo mazguose sutampa su kubinio polinomo antrųjų išvestinių reikmėmis tuose pačiuose mazguose, tai funkcija ***splg*** taškuose *x=t*, turi apskaičiuoti splaino reikšmes, kurios bus lygios $y = t^3 - 4t^2 + 1$.

Komandų lauke surinkime komandas ir pateikime jų atsakymus:

```
>> x=linspace(-1,5,7)
x = -1     0     1     2     3     4     5
>> y=x.^3-4*x.^2+1
y = -4     1    -2    -7    -8     1    26
>> m0=-14
m0 = -14
>> mn=22
mn = 22
>> fm=splin(x,y,m0,mn)
fm = -14.0000  -8.0000 -2.0000  4.0000 10.0000 16.0000 22.0000
>> y2i=6*x-8
y2i = -14    -8    -2     4    10    16    22.
```

Kaip ir reikėjo tikėtis, interpoliacinio splaino antrųjų išvestinių reikšmės, kurias apskaičiavo funkcija ***splin,*** sutapo su kubinio polinomo antrųjų išvestinių reikšmėmis.

Komandų lauke surinkime komandas:

```
>> t=-1:0.5:5
t = -1.0000 -0.5000 0  0.5000  1.0000  1.5000  2.0000  2.5000
     3.0000  3.5000  4.0000  4.5000  5.0000
>> gt=splg(x,y,fm,t)


gt = -4.0000 -0.1250  1.0000  0.1250 -2.0000 -4.6250 -7.0000 -8.3750
     -8.0000 -5.1250  1.0000 11.1250  26.0000
>> y=t.^3-4*t.^2+1
y =  -4.0000 -0.1250  1.0000  0.1250 -2.0000 -4.6250 -7.0000 -8.3750
     -8.0000 -5.1250  1.0000 11.1250 26.0000.
```

Matome, kad taškuose $x=t$, splaino reikšmės, kaip ir turi būti, sutampa su kubinio polinomo reikšmėmis.

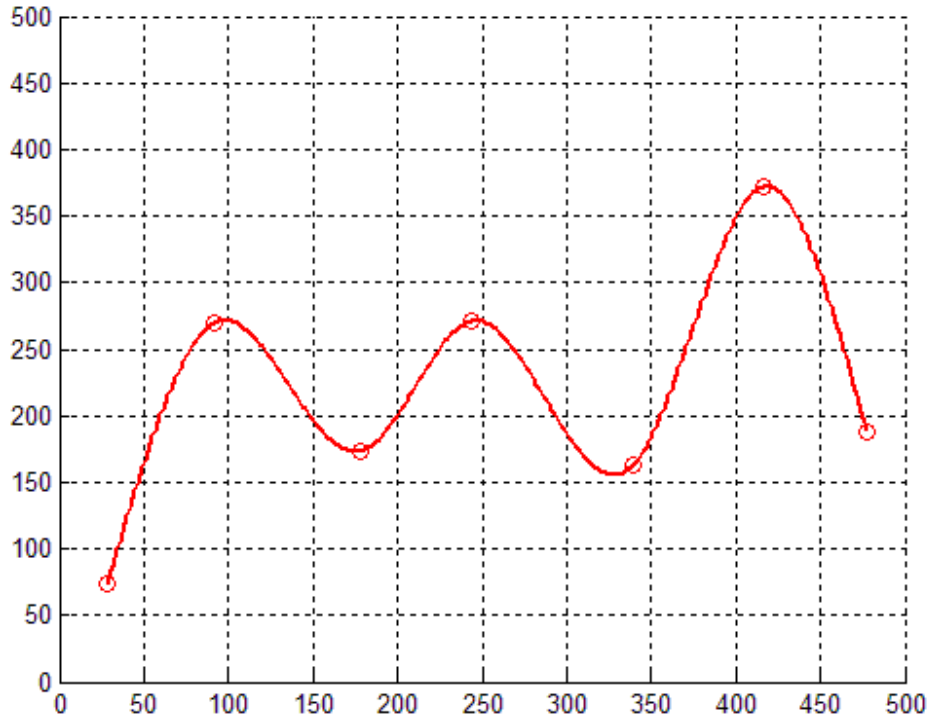
Sukurkime pagrindinę programą, kuri įgalintų vaizdžiai parodyti kubinio interpoliacinio splaino konstravimą. Tam tikslui grafiniame lange pažymėkime interpoliavimo taškus, įveskime kraštines sąlygas ir grafiškai pavaizduokime interpoliacinį splainą.

Žemiau pateikta tokia programa. Interpoliavimo taškai ekrane žymimi kairiuoju pelės klavišu, o paskutinysis taškas pažymimas dešiniuoju pelės klavišu. Antrosios kraštinės sąlygos įvedamos iš klaviatūros.

```
% Pagrindine programa, skirta demonstruoti
% interpoliacinį kubinį splainą, kai kraštinės sąlygos
% nusakomos nurodant antrųjų išvestinių reikšmes
clc
clear, close all
 hold on
 grid on
% Lango mastelis
x=[0 500];
y=[0 500];
axis([min(x) max(x) min(y) max(y)]);
%Interpoliavimo taškų žymėjimas
xt=[];
yt=[];
nn = 0;
% Ciklas taškams žymėti
disp('kairys mygtukas žymi taškus nuo 1 iki (n-1)')
disp('dešinysis - paskutinį n-ąjį tašką')
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    nn = nn+1;
    xt(1,nn)=xi;
    yt(1,nn)=yi;
end

% Kraštinių sąlygų įvedimas
m=input('įveskite kraštines sąlygas [m0 mn]')
% Interpoliacinio kubinio splaino apskaičiavimas
fm=splin(xt,yt,m(1),m(2));
% Taškai, kuriuose apskaičiuosime splaino reikšmes
t=linspace(xt(1),xt(end),300);
gt=splg(xt,yt,fm,t); %splaino reikšmės
hold on
plot(t,gt,'r','LineWidth',1.5) % brėžiamas splaino grafikas
```

2.16 paveikslėlyje pavaizduotas interpoliacinis kubinis splainas, sukonstruotas pateiktos pagrindinės programos pagalba.



2.16 pav. Interpolicinis kubinis splainas su natūraliomis kraštinėmis sąlygomis.