

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

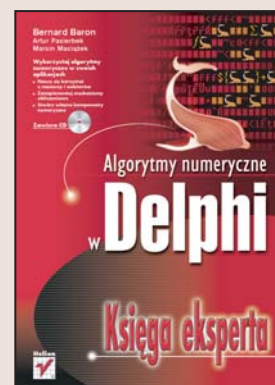
ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Algorytmy numeryczne w Delphi. Księga eksperta

Autorzy: Bernard Baron,
Artur Pasierbek, Marcin Maciążek
ISBN: 83-7361-951-8
Format: B5, stron: 544



Metody numeryczne są to sposoby rozwiązywania złożonych problemów matematycznych za pomocą narzędzi obliczeniowych udostępnianych przez popularne języki programowania. Jeden z najpopularniejszych języków – Pascal, będący podstawą języka ObjectPascal wykorzystywanego w Delphi, pozwala na bardzo łatwą implementację mechanizmów obliczeń numerycznych. Specyfika projektowania aplikacji w środowisku Delphi pozwala na utworzenie komponentów realizujących algorytmy numeryczne i stosowanie ich w wielu aplikacjach.

Książka „Algorytmy numeryczne w Delphi. Księga eksperta” przedstawia najczęściej wykorzystywane metody numeryczne wraz z przykładami ich implementacji w języku ObjectPascal. Każde zagadnienie jest omówione zarówno od strony teoretycznej, jak i praktycznej, co ułatwia jego zrozumienie i pozwala na modyfikacje zamieszczonych w książce kodów źródłowych.

- Typy, funkcje, klasy i procedury wykorzystywane w algorytmach numerycznych
- Algebra macierzy i równania liniowe
- Badanie funkcji
- Rozwiązywanie równań nieliniowych i wyznaczanie wartości własnych macierzy
- Układy równań różniczkowych liniowych i nieliniowych
- Przekształcenia Fouriera i Laplace’a

Niemal każdy problem obliczeniowy można rozwiązać za pomocą metod numerycznych. Nie musisz więc wymyślać ponownie koła – wystarczy, że poznasz opisane w tej książce algorytmy.



Spis treści

Zmiany w stosunku do poprzedniego wydania	9
---	---

Przedmowa	11
-----------------	----

Rozdział 1. Definicje typów, procedur, funkcji i klas dla zagadnień numerycznych 13

1.1. Organizacja biblioteki obliczeń numerycznych	14
1.2. Typ wariantowy	14
1.3. Predefiniowany typ liczb zespolonych	16
1.4. Definicja typu liczb zespolonych	17
1.5. Funkcje konwersji liczb rzeczywistych zespolonych na łańcuch i odwrotnie	18
1.6. Wektor	20
1.7. Macierz	21
1.8. Reprezentacja wektorów i macierzy za pomocą tablic	21
1.8.1. Przydzielanie i zwalnianie pamięci dla tablic jednowymiarowych	23
1.8.2. Przydzielanie i zwalnianie pamięci dla tablic dwuwymiarowych	24
1.9. Zapis i odczyt wektorów oraz macierzy w komponencie TStringGrid	25
1.10. Wzorcowe funkcje zapisu i odczytu plików macierzy	26

Rozdział 2. Algebra macierzy i równania liniowe27

2.1. Metoda bezpośredniego rozwiązywania układu równań macierzowych metodą eliminacji Gaussa	28
2.1.1. Skalowanie układu równań liniowych	32
2.2. Rozwiązywanie układu równań liniowych według algorytmu Crouta	34
2.3. Obliczanie macierzy odwrotnej metodą eliminacji Gaussa	39
2.4. Obliczanie macierzy odwrotnej metodą Crouta	43
2.5. Obliczanie wyznacznika macierzy kwadratowej	48
2.6. Wskaźnik uwarunkowania macierzy	50
2.7. Obliczanie wartości własnej macierzy kwadratowej A o największym module	52
2.8. Obliczanie wartości własnej macierzy $1-\alpha A$ o największym module	53
2.9. Rozwiązywanie układu równań liniowych metodą iteracji Jacobiego oraz Richardsona	55
2.10. Rozwiązywanie układu równań metodą Gaussa-Seidela oraz metodą nadrelaksacji	58
2.11. Pseudorozwiązanie układu nadokreślonego	60
2.12. Metoda najmniejszych kwadratów	66
2.13. Algorytm Crouta rozwiązywania rzadkich układów równań liniowych	68
2.14. Algorytmy iteracyjne Richardsona oraz Gaussa-Seidela dla macierzy rzadkich	78
Przykłady	85
Komponenty	85
Właściwości	85

Zdarzenia	86
Przykład 2.1. Obliczanie macierzy odwrotnej	88
Przykład 2.2. Rozwiązanie układów równań algebraicznych	95
Przykład 2.3. Rozwiązanie układów równań algebraicznych rzadkich	102

Rozdział 3. Praktyka badania funkcji 109

3.1. Całkowanie i różniczkowanie numeryczne	109
3.1.1. Ekstrapolacja iterowana Richardsona i Aitkena	109
3.1.2. Całkowanie numeryczne	116
3.1.3. Różniczkowanie numeryczne	125
3.1.4. Gradient funkcji wielu zmiennych	135
3.1.5. Jakobian funkcji wektorowej wielu zmiennych	136
3.1.6. Hesjan funkcji wielu zmiennych	137
3.2. Wybrane metody aproksymacji i interpolacji liniowej funkcji jednej zmiennej	138
3.2.1. Aproksymacja metodą najmniejszych kwadratów	139
3.2.2. Aproksymacja funkcji dyskretnej wielomianem	141
3.2.3. Aproksymacja układami funkcji ortogonalnych	141
3.2.4. Aproksymacja wielomianami ortogonalnymi	142
3.2.5. Implementacja metod aproksymacji	144
3.2.6. Interpolacja funkcji dyskretnej krzywą łamaną	159
3.2.7. Interpolacja wielomianem potęgowym Lagrange'a	160
3.2.8. Interpolacja funkcjami sklejanymi	160
3.2.9. Interpolacja funkcjami i wielomianami ortogonalnymi	162
3.2.10. Metody interpolacji w ramach klasy TInterpolation	165
3.3. Wybrane metody poszukiwania minimum funkcji wielu zmiennych metodami bezgradientowymi	180
3.3.1. Wyznaczenie minimum funkcji wielu zmiennych bezgradientową metodą poszukiwań prostych Hooke'a-Jeevesa	181
3.3.2. Bezgradientowa metoda „złotego podziału” poszukiwania minimum	184
3.3.3. Bezgradientowa metoda Powella poszukiwania minimum funkcji wielu zmiennych	192
3.4. Wybrane metody poszukiwania minimum funkcji wielu zmiennych metodami gradientowymi	196
3.4.1. Metoda ekspansji i kontrakcji geometrycznej z jednym testem badania współczynnika kroku przy poszukiwaniu minimum w kierunku	197
3.4.2. Metoda aproksymacji parabolicznej z jednym testem badania współczynnika kroku przy poszukiwaniu minimum w kierunku	201
3.4.3. Algorytm największego spadku	206
3.4.4. Zmodyfikowany algorytm Newtona	210
Przykłady	215
Komponenty	215
Przykład 3.1. Testowanie metod całkowania	216
Przykład 3.2. Testowanie procedur różniczkowania numerycznego	221
Przykład 3.3. Testowanie funkcji do wyznaczania macierzy Jacobiego funkcji wektorowej	225
Przykład 3.4. Testowanie funkcji do wyznaczania macierzy Hessego funkcji wielu zmiennych ..	229
Przykład 3.5. Testowanie metod klasy TApproximation	231
Przykład 3.6. Testowanie metod klasy TInterpolation	239
Przykład 3.7. Testowanie metod wyznaczania minimum funkcji	244

Rozdział 4. Równania nieliniowe, zera wielomianów, wartości własne macierzy251

4.1. Algorytmy rozwiązywania układów równań nieliniowych	252
4.1.1. Rozwiązywanie układów równań nieliniowych metodą Newtona	253
4.1.2. Rozwiązywanie układów równań nieliniowych metodą gradientową	256
4.1.3. Rozwiązywanie układu równań nieliniowych zmodyfikowaną metodą Newtona	260
4.1.4. Rozwiązywanie układów nieliniowych metodą iteracyjną	264
4.1.5. Pseudorozwiązania nieliniowego układu nadokreślonego metodą Hooke'a-Jeevesa	267
4.2. Wyznaczanie zer wielomianów metodami Bairstowa i Laguerre'a	270
4.2.1. Dzielenie wielomianów o współczynnikach rzeczywistych przez czynnik liniowy według algorytmu Hornera	270
4.2.2. Dzielenie wielomianu przez czynnik kwadratowy	272
4.2.3. Wyznaczanie dzielników wielomianu stopnia $N > 2$ w postaci trójmianu kwadratowego metodą Bairstowa	273
4.2.4. Wyznaczanie zer wielomianów o współczynnikach rzeczywistych	277
4.2.5. Wyznaczanie zer wielomianu metodą Laguerre'a	280
4.2.6. Wyznaczanie zer wielomianu metodą Laguerre'a	282
4.3. Wyznaczanie wartości własnych macierzy metodami Bairstowa i Laguerre'a	284
4.3.1. Wyznaczanie współczynników wielomianu charakterystycznego macierzy kwadratowej metodą Kryłowa	285
4.3.2. Wyznaczanie wartości własnych macierzy metodą Bairstowa	287
4.3.3. Wyznaczanie wartości własnych macierzy metodą Laguerre'a	290
4.4. Wyznaczanie zer funkcji jednej zmiennej metodą połowienia przedziału	291
Przykłady	293
Komponenty	293
Przykład 4.1. Testowanie metod rozwiązywania układu równań nieliniowych	294
Przykład 4.2. Testowanie metod rozwiązywania układu równań nieliniowych — cd.	295
Przykład 4.3. Wyznaczanie zer wielomianów o współczynnikach rzeczywistych zadanych z klawiatury za pomocą metod Laguerre'a oraz Bairstowa	300
Przykład 4.4. Wyznaczanie wartości własnej macierzy zadanej z klawiatury lub pliku	302
Przykład 4.5. Wyznaczanie zer i ekstremum funkcji Bessela rzędu N	305

Rozdział 5. Układy zwyczajnych równań różniczkowych nieliniowych309

5.1. Układ równań różniczkowych jako klasa programowania obiektowego	310
5.1.1. Definicje typów do zadawania układu równań różniczkowych nieliniowych	311
5.1.2. Definicja klasy prototypowej dla klas implementujących rozwiązywanie układu równań różniczkowych	312
5.1.3. Definicja klasy prototypowej dla klas potomnych dotyczących rozwiązywania układu równań różniczkowych nieliniowych	318
5.1.4. Aproksymacja dyskretnych wartości wektorów stanu	319
5.1.5. Funkcje pomocnicze do działania na wektorach stanu	322
5.2. Metody Rungego-Kutty	323
5.3. Rozwiązywanie układu równań różniczkowych zwyczajnych metodą Rungego-Kutty z automatycznym doбором kroku całkowania	327
5.4. Metody Fehlberga	332

5.5. Rozwiązanie układu równań różniczkowych nieliniowych zwyczajnych metodą Fehlbberga z automatycznym doбором kroku całkowania	340
5.6. Rozwiązanie układu równań różniczkowych nieliniowych zwyczajnych metodą Dormanda-Prince'a z automatycznym doбором kroku całkowania	344
5.7. Wielokrokowa metoda rozwiązywania układu równań różniczkowych nieliniowych z członem przewidywania Adamsa-Bashfortha oraz członem korekcyjnym Adamsa-Multona z automatycznym doбором kroku i rzędu	349
5.7.1. Algorytm Adamsa-Bashfortha	349
5.7.2. Algorytm Adamsa-Multona	351
5.7.3. Algorytmy przewidywania i korekcji wyrażone przez macierz Nordsiecka	354
5.7.4. Faza wstępna obliczeń	363
5.7.5. Metody klasy TAdamsMultonAbstract i TAdamsMulton, realizujące algorytm Adamsa-Multona	368
5.8. Rozwiązywanie układu równań nieliniowych metodą sztywno stabilnych algorytmów Geara	374
5.9. Metoda Gragga z ekstrapolacją Bulirsch-Stoera	386
Przykłady	394
Komponenty	394
Przykład 5.1. Rozwiązywanie układów równań różniczkowych drugiego rzędu	395
Przykład 5.2. Zastosowanie klasy TRoRoNI do rozwiązywania układów równań różniczkowych nieliniowych w ramach pewnej klasy	402
Przykład 5.3. Wahadło matematyczne	408

Rozdział 6. Układy równań różniczkowych liniowych

o stałych współczynnikach413

6.1. Równania różnicowe dla różnych aproksymacji funkcji wymuszających	418
6.1.1. Wymuszenie aproksymowane funkcjami przedziałami stałymi	418
6.1.2. Wymuszenie aproksymowane funkcjami przedziałami liniowymi	420
6.1.3. Wymuszenie aproksymowane wielomianem stopnia drugiego	422
6.1.4. Dobór kroku całkowania T ze względu na dobór górnej granicy błędu obliczania macierzy e^{AT} oraz ze względu na numeryczną stabilność rozwiązania	425
6.2. Definicja typów dla liniowych równań różniczkowych	427
6.3. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami stałymi	429
6.4. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami liniowymi	431
6.5. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami kwadratowymi	433
Przykłady	435
Komponenty	435
Przykład 6.1. Testowanie metod rozwiązywania układu równań różniczkowych liniowych ..	435
Przykład 6.2. Testowanie metod rozwiązywania układu równań różniczkowych liniowych zdefiniowanych wewnątrz pewnej klasy	440

Rozdział 7. Praktyka przekształceń Fouriera449

7.1. Dyskretna transformacja Fouriera według algorytmu Hornera	455
7.2. Szybkie przekształcenie Fouriera według algorytmu Cooley-Tukeya	457
7.3. Szybkie przekształcenie Fouriera według algorytmu Sande'a-Tukeya	466
7.4. Wyznaczanie współczynników zespolonego szeregu Fouriera dla dowolnej funkcji okresowej ..	470
7.5. Obliczanie odwrotnej transformacji Fouriera dla dowolnej transformaty	471

Przykłady	474
Komponenty	474
Przykład 7.1. Obliczanie zespolonych współczynników szeregu Fouriera	475
Przykład 7.2. Obliczanie odwrotnej transformacji Fouriera	479
Przykład 7.3. Obliczanie zespolonych współczynników szeregu Fouriera w ramach pewnej klasy	483
Rozdział 8. Praktyka przekształceń Laplace'a	487
8.1. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasowej z zastosowaniem szeregów Fouriera	488
8.2. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasowej z zastosowaniem szeregów Laguerre'a	494
8.3. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasowej według algorytmu Valsa	498
8.4. Obliczanie transformacji odwrotnej Laplace'a funkcji wymiernej na podstawie jej pozostałości w biegunach	502
8.4.1. Definicja klasy do obliczania odwrotnej transformacji Laplace'a funkcji wymiernej na podstawie jej pozostałości w biegunach	505
Przykłady	510
Komponenty	510
Przykład 8.1. Wyznaczanie odwrotnej transformacji Laplace'a funkcji operatorowych zgodnie ze wzorcami funkcji	511
Przykład 8.2. Zastosowanie transformacji odwrotnej Laplace'a dla funkcji wymiernych	516
Bibliografia	523
Skorowidz	525

Rozdział 6.

Układy równań różniczkowych liniowych o stałych współczynnikach

Zadany jest układ N równań różniczkowych liniowych niejednorodnych:

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^N a_{ij} x_j(t) + \sum_{j=1}^W b_{ij} u_j(t) \quad (i = 1, 2, \dots, N), \quad (6.1)$$

gdzie współczynniki a_{ij} oraz b_{ij} są rzeczywiste. Układ ten można zapisać w postaci macierzowej:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{Ax}(t) + \mathbf{Bu}(t), \quad (6.2)$$

gdzie:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ \vdots \\ x_N(t) \end{bmatrix}; \quad (6.2a)$$

$$\frac{d\mathbf{x}(t)}{dt} = \begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \\ \vdots \\ \vdots \\ \frac{dx_N(t)}{dt} \end{bmatrix}; \quad (6.2b)$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}; \quad (6.2c)$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1W} \\ b_{21} & b_{22} & \dots & b_{2W} \\ \dots & \dots & \dots & \dots \\ b_{N1} & b_{N2} & \dots & b_{NW} \end{bmatrix}; \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_w(t) \end{bmatrix}. \quad (6.2d)$$

Na człony niejednorodny układu (6.1) składa się W wymuszeń $u_j(t)$ ($j = 1, 2, \dots, W$) występujących ze współczynnikami b_{ij} macierzy prostokątnej $\mathbf{B}$. W teorii równania (6.2) centralną rolę odgrywa funkcja wykładnicza $e^{\mathbf{A}t}$ macierzy kwadratowej $\mathbf{A}$ przemnożonej przez zmienną niezależną t , zdefiniowaną szeregiem macierzowym [7]:

$$e^{\mathbf{A}t} = \mathbf{1} + \mathbf{A}t + \frac{1}{2!}(\mathbf{A}t)^2 + \dots + \frac{1}{k!}(\mathbf{A}t)^k + \dots = \sum_{k=0}^{\infty} \frac{(\mathbf{A}t)^k}{k!}. \quad (6.3)$$

Szereg macierzowy (6.3) jest równoważny N^2 zwykłym skalarnym szeregom potęgowym:

$$\delta_{ij} + (\mathbf{A}t)_{ij} + \frac{1}{2!}\{(\mathbf{A}t)^2\}_{ij} + \dots + \frac{1}{k!}\{(\mathbf{A}t)^k\}_{ij} + \dots, \quad (i, j = 1, 2, \dots, N).$$

Do zrozumienia konstrukcji całki ogólnej równania (6.2) niezbędne będą następujące własności funkcji wykładniczej $e^{\mathbf{A}t}$:

1. Jeżeli $t = 0$, to zgodnie z definicją (6.3)

$$e^{\mathbf{A}0} = \mathbf{1} \text{ (macierz jednostkowa } N \cdot N\text{-wymiarowa)}. \quad (6.4)$$

2. Jeżeli macierz $\mathbf{A}$ komutuje z macierzą $\mathbf{B}$, a więc $\mathbf{AB} = \mathbf{BA}$, to:

$$e^{\mathbf{A}t} \cdot e^{\mathbf{B}t} = e^{(\mathbf{A}+\mathbf{B})t}. \quad (6.5)$$

3. Ponieważ na mocy własności (6.5) $e^{\mathbf{A}t} e^{-\mathbf{A}t} = e^{(\mathbf{A}-\mathbf{A})t} = \mathbf{1}$, więc macierz odwrotna macierzy $e^{\mathbf{A}t}$ ma postać:

$$[e^{\mathbf{A}t}]^{-1} = e^{-\mathbf{A}t}. \quad (6.6)$$

4. Różniczkując obie strony równania macierzowego (6.3) ze względu na t oraz wyłączając wspólny czynnik $\mathbf{A}$ z wyrazów szeregu nieskończonego, otrzymuje się:

$$\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A}. \quad (6.7)$$

5. Mnożąc lewostronnie lub prawostronnie równanie macierzowe (6.7) przez $\mathbf{A}^{-1}$ (macierz odwrotna macierzy $\mathbf{A}$), a następnie całkując tak otrzymywane równania ze względu na t od t_1 do t_2 , otrzymuje się:

$$\int_{t_1}^{t_2} e^{At} dt = \mathbf{A}^{-1} (e^{At_2} - e^{At_1}) = (e^{At_2} - e^{At_1}) \mathbf{A}^{-1}. \quad (6.8)$$

Do rozwiązania układu równań różniczkowych liniowych (6.2) można zastosować metodę uziemiennienia stałych. W tym celu najpierw rozpatruje się przypadek, gdy $\mathbf{u}(t) \equiv 0$, co oznacza, że równanie (6.2) jest jednorodne

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t). \quad (6.9)$$

Łatwo wykazać, że całka ogólna równania jednorodnego (6.9) ma postać:

$$\mathbf{x}(t) = e^{At} \mathbf{y}, \quad (6.10)$$

gdzie $\mathbf{y}$ jest wektorem N -wymiarowym o składowych stałych.

Istotnie z własności (6.7) wynika

$$\frac{d\mathbf{x}(t)}{dt} = \frac{d}{dt} (e^{At} \mathbf{y}(t)) = \mathbf{A}e^{At} \mathbf{y} = \mathbf{A}\mathbf{x}(t). \quad (6.11)$$

Zgodnie z metodą uziemiennienia stałych przyjmuje się dalej, że wektor $\mathbf{y}$ jest funkcją zmiennej t , co daje:

$$\mathbf{x}(t) = e^{At} \mathbf{y}(t), \quad (6.12)$$

a następnie podstawia się wyrażenie (6.12) do równania niejednorodnego (6.2), uwzględniając własność (6.7)

$$\mathbf{A}e^{At} \mathbf{y}(t) + e^{At} \frac{d\mathbf{y}(t)}{dt} = \mathbf{A}e^{At} \mathbf{y}(t) + \mathbf{B}\mathbf{u}(t). \quad (6.13)$$

Upraszczając równanie (6.13) o człon $\mathbf{A}e^{At} \mathbf{y}(t)$ oraz mnożąc je lewostronnie przez macierz e^{-At} , otrzymuje się na mocy własności (6.6)

$$\frac{d\mathbf{y}(t)}{dt} = e^{-At} \mathbf{B}\mathbf{u}(t). \quad (6.14)$$

Całkując równanie (6.14) ze względu na t od t_0 do t , otrzymuje się:

$$\mathbf{y}(t) = \mathbf{y}(t_0) + \int_{t_0}^t e^{-A\tau} \mathbf{B}\mathbf{u}(\tau) d\tau. \quad (6.15)$$

Jeżeli zadany jest wektor wartości początkowych $\mathbf{x}(t_0)$, to odpowiadający mu wektor $\mathbf{y}(t_0)$ można wyznaczyć z równania (6.12), stosując własność (6.6):

$$\mathbf{y}(t_0) = e^{-At_0} \mathbf{x}(t_0). \quad (6.16)$$

Uwzględniając równanie (6.15) wraz z podstawieniem (6.16) w równaniu (6.12), otrzymuje się następujące rozwiązanie równania (6.2):

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)} \mathbf{x}(t_0) + e^{\mathbf{A}t} \int_{t_0}^t e^{-\mathbf{A}\tau} \mathbf{B} \mathbf{u}(\tau) d\tau. \quad (6.17)$$

Równanie (6.17) nie nadaje się do bezpośredniego obliczenia numerycznego. Rozwiązanie dokładne (6.17) równania (6.2) można jednak wykorzystać w metodzie krokowej, zastępując to równanie równaniem różnicowym, przyjmując $t_0 = kT$ i $t = (k+1)T$:

$$\mathbf{x}[(k+1)T] = e^{\mathbf{A}T} \mathbf{x}(kT) + e^{\mathbf{A}(k+1)T} \int_{kT}^{(k+1)T} e^{-\mathbf{A}\tau} \mathbf{B} \mathbf{u}(\tau) d\tau. \quad (6.18)$$

W obliczaniu całek (6.18) mogą wystąpić trudności związane z występowaniem ujemnych i dużych co do modułu wartości własnych macierzy $\mathbf{A}$. Ze względu na możliwość takiego przypadku należy aproksymować funkcję wektorową wymuszającą $\mathbf{u}(t)$, nie zmieniając jądra $e^{\mathbf{A}t}$ w całce równania (6.18).

Niech zachodzi przypadek ogólny, dla którego macierz $\mathbf{A}$ ma dzielniki elementarne:

$$(\lambda - \lambda_1)^{p_1}, (\lambda - \lambda_2)^{p_2}, \dots, (\lambda - \lambda_s)^{p_s},$$

gdzie wśród wartości własnych $\lambda_1, \lambda_2, \dots, \lambda_s$ macierzy $\mathbf{A}$ będących, zgodnie z definicją, zerami wielomianu charakterystycznego macierzy $\mathbf{A}$

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0,$$

mogą być liczby jednakowe; $1 \leq p_n \leq N$, przy czym $p_1 + p_2 + \dots + p_s = M$. Dowodzi się, że w takim przypadku istnieje taka macierz nieosobliwa $\mathbf{S}$, że

$$\mathbf{A} = \mathbf{S}^{-1} \mathbf{C} \mathbf{S}, \quad (6.19)$$

gdzie macierz $\mathbf{C}$ jest macierzą quasi-diagonalną, zwaną kanoniczną macierzą Jordana [30].

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_{p_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & \mathbf{I}_{p_2}(\lambda_2) & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ 0 & 0 & \dots & \mathbf{I}_{p_s}(\lambda_s) \end{bmatrix}; \quad (6.20)$$

$$\mathbf{I}_{p_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 0 & 0 & \dots & 0 & 0 \\ 1 & \lambda_i & 0 & \dots & 0 & 0 \\ 0 & 1 & \lambda_i & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & \lambda_i & 0 \\ 0 & 0 & 0 & \dots & 1 & \lambda_i \end{bmatrix}.$$

Stosując transformację (6.19), funkcję wykładniczą e^{At} można przekształcić następująco:

$$e^{At} = e^{(S^{-1}CS)t} = e^{S^{-1}(Ct)S} = S^{-1} e^{Ct} S. \quad (6.21)$$

Ponieważ macierz C jest quasi-diagonalna, to:

$$e^{Ct} = \begin{bmatrix} e^{1_{p_1}(\lambda_1)t} & 0 & 0 \\ 0 & e^{1_{p_2}(\lambda_2)t} & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & e^{1_{p_s}(\lambda_s)t} \end{bmatrix}. \quad (6.22)$$

Zgodnie z definicją macierzowej funkcji wykładniczej oraz macierzy (6.20) zachodzi [30]:

$$e^{1_{p_i}(\lambda_i)t} = \begin{bmatrix} e^{\lambda_i t} & 0 & 0 & \dots & 0 \\ te^{\lambda_i t} & e^{\lambda_i t} & 0 & \dots & 0 \\ \frac{t^2}{2!} e^{\lambda_i t} & te^{\lambda_i t} & e^{\lambda_i t} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \frac{t^{p_i-1}}{(p_i-1)!} e^{\lambda_i t} & \frac{t^{p_i-2}}{(p_i-2)!} e^{\lambda_i t} & \frac{t^{p_i-3}}{(p_i-3)!} e^{\lambda_i t} & \dots & e^{\lambda_i t} \end{bmatrix}. \quad (6.23)$$

Wzory (6.17), (6.21) i (6.22) określają strukturę rozwiązania równania różniczkowego (6.2), a w szczególności jego związek z wartościami własnymi λ_i występującymi w kombinacjach funkcji $e^{\lambda_i t}$ przemnożonych przez wielomiany $P_i(t)$ stopnia nie większego niż p_i-1 , gdzie p_i jest stopniem dzielnika elementarnego odpowiadającego wartości własnej λ_i , tj. $P_i(t)e^{\lambda_i t}$. Załóżmy w ogólnym przypadku, że wartości własne λ_i macierzy A są zespolone

$$\lambda_i = \alpha_i + j\beta_i \quad (i = 1, 2, \dots, N). \quad (6.24)$$

Jeżeli $\text{Re}\{\lambda_i\} = \alpha_i > 0$, to odpowiednie składniki rozwiązania $P_i(t)$ wzrastają wykładniczo z członem wielomianowym $P_i(t)$, gdy czas t wzrasta. Jeżeli $\alpha_i < 0$, to odpowiednie składniki rozwiązania $P_i(t)e^{\lambda_i t}$ maleją, gdy czas t wzrasta.

W każdym przypadku, jeśli $\text{Im}\{\lambda_i\} = \beta_i \neq 0$, to — jak wiadomo — λ_i tworzy zespoloną parę sprzężoną z odpowiednią wartością własną λ_i^* , co odpowiada składnikowi rozwiązania sinusoidalnemu z wagą wykładniczą $e^{\lambda_i t}$ i wielomianową $P_i(t)$:

$$P_i(t)e^{\alpha_i t} \sin \beta_i t. \quad (6.25)$$

6.1. Równania różnicowe dla różnych aproksymacji funkcji wymuszających

Do numerycznego rozwiązania układu równań różniczkowych liniowych (6.2) można wykorzystać równanie różnicowe (6.18), przyjmując różną aproksymację funkcji wymuszającej $\mathbf{u}(t)$. W niniejszym opracowaniu podane będą konstrukcje tych algorytmów dla trzech przypadków, a mianowicie dla aproksymacji funkcji wymuszającej w postaci funkcji przedziałami stałej, liniowej i kwadratowej.

6.1.1. Wymuszenie aproksymowane funkcjami przedziałami stałymi

Niech wymuszenie wektorowe $\mathbf{u}(t)$ jest dane w postaci funkcji przedziałami stałej takiej, że:

$$\mathbf{u}(t) = \mathbf{u}(kT) \quad \text{dla } kT \leq t \leq (k+1)T, k = 0, 1, 2, \dots \quad (6.26)$$

W takim przypadku, wykonując całkowanie w równaniu różnicowym (6.18) z uwzględnieniem wzoru (6.8), otrzymuje się [7]:

$$\begin{aligned} \int_{kT}^{(k+1)T} e^{-\mathbf{A}\tau} \mathbf{B} \mathbf{u}(\tau) d\tau &= -e^{-\mathbf{A}\tau} \Big|_{kT}^{(k+1)T} \mathbf{A}^{-1} \mathbf{B} \mathbf{u}(kT) = \\ &= \left(-e^{-\mathbf{A}(k+1)T} + e^{-\mathbf{A}kT} \right) \mathbf{A}^{-1} \mathbf{B} \mathbf{u}(kT) \end{aligned} \quad (6.27)$$

Po umieszczeniu powyższego wyniku całkowania w równaniu (6.18) otrzymuje się:

$$\begin{aligned} \mathbf{x}[(k+1)T] &= e^{\mathbf{A}T} \mathbf{x}(kT) + e^{\mathbf{A}(k+1)T} \left(-e^{-\mathbf{A}(k+1)T} + e^{-\mathbf{A}kT} \right) \mathbf{A}^{-1} \mathbf{B} \mathbf{u}(kT) = \\ &= e^{\mathbf{A}T} \mathbf{x}(kT) + (e^{\mathbf{A}T} - \mathbf{1}) \mathbf{A}^{-1} \mathbf{B} \mathbf{u}(kT) \end{aligned} \quad (6.28)$$

gdzie: $\mathbf{1}$ — macierz jednostkowa.

W równaniu różnicowym (6.28) celowym jest, ze względu na minimum operacji numerycznych, obliczać macierz $(e^{\mathbf{A}T} - \mathbf{1}) \mathbf{A}^{-1}$, nie wykonując pomocniczych obliczeń macierzy $e^{\mathbf{A}T}$ oraz $\mathbf{A}^{-1}$, lecz wykorzystując równość:

$$(e^{\mathbf{A}T} - \mathbf{1}) \mathbf{A}^{-1} = T \sum_{n=0}^{\infty} \frac{(\mathbf{A}T)^n}{(n+1)!} \quad (6.29)$$

wynikającą z definicji (6.3).

Zatem po uwzględnieniu równania (6.29) oraz oznaczenia macierzy:

$$\mathbf{F} = e^{\mathbf{A}T} = T \sum_{n=0}^{\infty} \frac{(\mathbf{A}T)^n}{n!} \quad (6.30)$$

$$\mathbf{G}_0 = \left[\sum_{n=0}^{\infty} \frac{(\mathbf{A}T)^n}{(n+1)!} \right] \mathbf{B}T \quad (6.31)$$

i wektorów

$$\mathbf{x}(k) = \mathbf{x}(kT); \quad \mathbf{u}(k) = \mathbf{u}(kT) \quad (6.32)$$

formuła rekurencyjna (6.28) przyjmie postać:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}_0\mathbf{u}(k). \quad (6.33)$$

Nie istnieje więc potrzeba obliczania macierzy odwrotnej $\mathbf{A}^{-1}$, jak by to wynikało z równania (6.28). Mając na uwadze dalszą minimalizację operacji numerycznych, należy zauważyć, że formowanie macierzy $\mathbf{F}$ i $\mathbf{G}_0$ (wzory (6.30) i (6.31)) należy prowadzić równoległe ze względu na wspólne elementy $(\mathbf{A}T)$ występujące w szeregach. Równanie różnicowe (6.33) daje więc formułę rekurencyjną, którą można łatwo zaprogramować na komputerze, co pokazane będzie w dalszych punktach.

Stosując wzór rekurencyjny (6.33) do rozwiązania numerycznego równania różniczkowego (6.2), odpowiadający aproksymacji wymuszeń funkcjami przedziałami stałymi, należy w pierwszej kolejności wygenerować macierze $\mathbf{F}$ i $\mathbf{G}_0$, określone wzorami (6.30) i (6.31). Blok funkcyjny generujący te macierze może mieć postać:

```
function FmTemp1(var A, B, F, G1: TMatrixF; T, eps, EpsR: TFloat;
  N, W: Integer): Integer;
  // Formowanie macierzy pomocniczych F, G1:
  // A, B — macierze układu równań różniczkowych
  // dX/dt = A*X+B*U,
  // N — rząd macierzy A,
  // W — liczba kolumn macierzy B,
  // T — wybrany krok całkowania,
  // eps — górna granica błędu przybliżenia macierzy F i G1,
  // EpsR — błąd wyznaczenia największej co do modułu wartości
  // własnej macierzy F
var
  K, Error: Integer;
  S, S1, NormAT, teta, MWA: TFloat;
  AX, AY, at, BX, BT: TMatrixF;
begin
  Result := 0;
  SetLength(at, N + 1, N + 1);
  SetLength(AX, N + 1, N + 1);
  SetLength(BX, N + 1, N + 1);
  SetLength(AY, N + 1, N + 1);
  SetLength(BT, N + 1, W + 1);
  try
    mMulR(at, A, T);
    mOne(AX);
    NormAT := mNorm(at);
    K := 0;
    S := 1;
    S1 := 1;
    teta := NormAT / (1 - NormAT);
    mClone(F, AX);
    mClone(BX, AX);
    repeat
      Inc(K);
      mMul(AY, AX, at);
      S := S / K;
      mMulr(AX, AY, S);
```

```

    mAdd(F, F, AX);
    S1 := S1 / (K + 1);
    mMulr(AX, AY, S1);
    mAdd(BX, BX, AX);
    mClone(AX, AY);
    teta := teta * NormAT / (K + 1)
until teta < eps;
Error := mEigenValue(MWA, F, EpsR, 1000);
if MWA >= 1.05 then
    Result := 16;
if Error < 0 then
    Result := 17;
mMulr(BT, B, T);
mMul(G1, BX, BT);
finally
    at := nil;
    BT := nil;
    AX := nil;
    BX := nil;
    AY := nil;
end
end{FmTemp1 };

```

6.1.2. Wymuszenie aproksymowane funkcjami przedziałami liniowymi

Zakładamy, że wymuszenie $\mathbf{u}(t)$ jest funkcją ciągłą przedziałami liniową taką, że:

$$\mathbf{u}(\tau) = \mathbf{u}(kT) + \frac{1}{T} [\mathbf{u}((k+1)T) - \mathbf{u}(kT)](\tau - kT) = \mathbf{f}_1 + \mathbf{f}_2 \tau \quad (6.34)$$

dla

$$kT \leq \tau < (k+1)T, \quad k = 0, 1, 2, \dots$$

gdzie:

$$\mathbf{f}_1 = \mathbf{u}(kT) - k[\mathbf{u}((k+1)T) - \mathbf{u}(kT)]; \quad \mathbf{f}_2 = \frac{1}{T} [\mathbf{u}((k+1)T) - \mathbf{u}(kT)]. \quad (6.34a)$$

Wykonując w takim przypadku całkowanie przez części w równaniu różnicowym (6.18) z uwzględnieniem wzoru (6.8), otrzymuje się:

$$\begin{aligned}
 \int_{kT}^{(k+1)T} e^{A\tau} \mathbf{B} \mathbf{u}(\tau) d\tau &= -\mathbf{A}^{-1} e^{-A\tau} \mathbf{B} (\mathbf{f}_1 + \mathbf{f}_2 \tau) \Big|_{kT}^{(k+1)T} + \int_{kT}^{(k+1)T} \mathbf{A}^{-1} e^{-A\tau} \mathbf{B} \mathbf{f}_2 d\tau = \\
 &= \mathbf{A}^{-1} e^{-AkT} \left\{ \left[\mathbf{B} + \mathbf{A}^{-1} (e^{-AkT} - \mathbf{1}) \mathbf{B} \frac{1}{T} \right] \mathbf{u}(kT) + \right. \\
 &\quad \left. + \left[e^{-AT} \mathbf{B} - \mathbf{A}^{-1} (e^{-AT} - \mathbf{1}) \mathbf{B} \frac{1}{T} \right] \mathbf{u}((k+1)T) \right\}.
 \end{aligned}$$

Po uwzględnieniu powyższego wyniku całkowania oraz oznaczenia (6.32) równanie różnicowe (6.18) przyjmie postać:

$$\begin{aligned} \mathbf{x}(k+1) = e^{AT} \mathbf{x}(k) + \mathbf{A}^{-1} \left[e^{AT} - (e^{AT} - \mathbf{1}) \mathbf{A}^{-1} \frac{1}{T} \right] \mathbf{B} \mathbf{u}(k) + \\ + \mathbf{A}^{-1} \left[(e^{AT} - \mathbf{1}) \mathbf{A}^{-1} \frac{1}{T} - \mathbf{1} \right] \mathbf{B} \mathbf{u}(k+1). \end{aligned} \quad (6.35)$$

Uwzględniając wzory (6.30) i (6.29), równanie rekurencyjne (6.35) można przekształcić do postaci:

$$\mathbf{x}(k+1) = \mathbf{F} \mathbf{x}(k) + \mathbf{G}_1 \mathbf{u}(k) + \mathbf{H} \mathbf{u}(k+1), \quad (6.36)$$

gdzie:

$$\mathbf{G}_1 = \mathbf{A}^{-1} \left[e^{AT} - (e^{AT} - \mathbf{1}) - \mathbf{A}^{-1} \frac{1}{T} \right] \mathbf{B} = \left[\sum_{n=0}^{\infty} \frac{(AT)^n}{n!(n+2)} \right] (\mathbf{B} \mathbf{T}); \quad (6.37)$$

$$\mathbf{H} = \mathbf{A}^{-1} \left[(e^{AT} - \mathbf{1}) \mathbf{A}^{-1} - \mathbf{1} \right] \mathbf{B} = \left[\sum_{n=0}^{\infty} \frac{(AT)^n}{(n+2)!} \right] (\mathbf{B} \mathbf{T}), \quad (6.38)$$

natomiast macierz $\mathbf{F}$ wyraża się wzorem (6.30).

Równanie rekurencyjne (6.36) daje więc algorytm wyznaczania rozwiązania równania różniczkowego w postaci (6.2). W obliczeniach komputerowych należy zauważyć, że wyznaczanie macierzy $\mathbf{F}$, $\mathbf{G}_1$ i $\mathbf{H}$ zgodnie ze wzorami (6.30), (6.37) i (6.38) należy prowadzić równolegle ze względu na wspólne elementy $(AT)^n$ występujące w szeregach macierzowych tych wzorów, co minimalizuje liczbę operacji numerycznych. W przypadku stosowania wzoru rekurencyjnego (6.36) niezbędne jest wygenerowanie macierzy $\mathbf{F}$, $\mathbf{G}_1$ i $\mathbf{H}$ (wzory (6.30), (6.37) i (6.38)), co można zrealizować w następującym bloku funkcyjnym:

```
function FmTemp2(var A, B, F, G2, H: TMatrixF; T, eps, EpsR: TFloat;
  N, W: Integer): Integer;
// Formowanie macierzy pomocniczych F, G2, H:
// A, B — macierze układu równań różniczkowych dX/dt = A*X+B*U,
// N — rząd macierzy A i F,
// W — liczba kolumn macierzy B, G2, H,
// T — wybrany krok całkowania,
// eps — górna granica błędu przybliżenia macierzy F, G2, H,
// EpsR — błąd wyznaczenia największej co do modułu wartości
// własnej macierzy F
var
  K, Error: Integer;
  SS, S1, S2, NormAT, teta, MWA: TFloat;
  AX, AY, at, AG, AH, BT: TMatrixF;
begin
  Result := 0;
  SetLength(at, N + 1, N + 1);
  SetLength(AX, N + 1, N + 1);
  SetLength(AY, N + 1, N + 1);
  SetLength(AH, N + 1, N + 1);
  SetLength(AG, N + 1, N + 1);
  SetLength(BT, N + 1, W + 1);
```

```

try
  mMulR(at, A, T);
  mOne(AX);
  NormAT := mNorm(at);
  K      := 0;
  SS     := 1;
  S1     := 0.5;
  S2     := 0.5;
  teta   := NormAT / (1 - NormAT);
  mOne(F);
  mMulr(AG, AX, 0.5);
  mClone(AH, AG);
  repeat
    Inc(K);
    mMul(AY, AX, at);
    SS := SS / K;
    mMulr(AX, AY, SS);
    mAdd(F, F, AX);
    S1 := S1 * (K + 1) / ((K + 2) * K);
    mMulr(AX, AY, S1);
    mAdd(AG, AG, AX);
    S2 := S2 / (K + 2);
    mMulr(AX, AY, S2);
    mAdd(AH, AH, AX);
    mClone(AX, AY);
    teta := teta * NormAT / (K + 1)
  until teta < eps;
  Error := mEigenValue(MWA, F, EpsR, 1000);
  if MWA >= 1.05 then
    Result := 16;
  if Error <> 0 then
    Result := 17;
  mMulr(BT, B, T);
  mMul(G2, AG, BT);
  mMul(H, AH, BT);
finally
  at := nil;
  BT := nil;
  AX := nil;
  AY := nil;
  AG := nil;
  AH := nil;
end
end {FmTemp2 };

```