helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

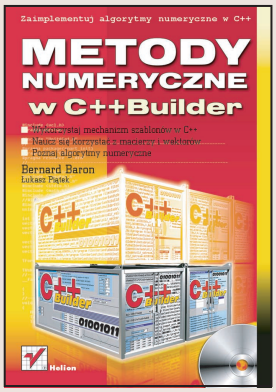
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Metody numeryczne w C++Builder



Autorzy: Bernard Baron, Łukasz Piątek
ISBN: 83-7361-544-X
Format: B5, stron: 552

Metody numeryczne są to sposoby rozwiązywania złożonych problemów matematycznych za pomocą narzędzi obliczeniowych udostępnianych przez popularne języki programowania. Jeden z najpopularniejszych języków – C++, chociaż nie był projektowany z myślą o zastosowaniu go w obliczeniach numerycznych, posiada mechanizmy, które umożliwiają stosunkowo łatwą implementację algorytmów obliczeniowych. Dzięki uniwersalności mechanizmu szablonów programista może tworzyć procedury numeryczne, w których da się określić precyzję obliczeń zmiennoprzecinkowych. Procedury stworzone w C++ nadają się do przeprowadzania obliczeń zarówno w dziedzinie liczb rzeczywistych, jak i zespolonych.

Książka „Metody numeryczne w C++Builder" przedstawia najczęściej wykorzystywane algorytmy numeryczne wraz z przykładami ich implementacji w języku C++. Każde zagadnienie jest omówione zarówno od strony teoretycznej, jak i praktycznej, co ułatwia jego zrozumienie i pozwala na modyfikacje zamieszczonych w książce kodów źródłowych. Książka zawiera również opis zagadnień związanych z językiem C++, niezbędnych do poznania i prawidłowego wykorzystywania biblioteki obliczeń numerycznych.

- Algebra macierzy i równania liniowe
- Całkowanie i różniczkowanie numeryczne
- Wybrane algorytmy interpolacji i aproksymacji
- Wyznaczanie minimów funkcji
- Rozwiązywanie równań nieliniowych i wyznaczanie wartości własnych macierzy
- Układy równań różniczkowych liniowych i nieliniowych
- Przekształcenia Fouriera i Laplace'a

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Przedmowa 7

Rozdział 1. Definicje typów, funkcji, klas i wzorców dla zagadnień numerycznych .. 9

1.1. Zastosowanie wzorców C++ w bibliotece obliczeń numerycznych 10
1.2. Definicja wzorca klasy liczb zespolonych 13
1.3. Organizacja biblioteki obliczeń numerycznych 15
1.4. Funkcje konwersji liczb rzeczywistych zespolonych na łańcuch i odwrotnie 16
1.5. Użycie wzorca klasy vector do implementacji wektorów w języku C++ 18
1.5.1. Operacje na wektorach zdefiniowanych na bazie konteneru vector 20
1.6. Macierz jako wektor wektorów 21
1.7. Zapis i odczyt wektorów oraz macierzy na komponencie TStringGrid 24
1.8. Funkcje wzorcowe do zapisu i odczytu plików macierzy 24
1.9. Wykorzystanie funkcji matematycznych zawartych w bibliotece math.h 25
1.10. Przekazywanie wskaźników funkcji do procedur implementujących algorytmy obliczeń numerycznych 27
1.11. DynamicArray i wzorzec valarray jako alternatywa dla wzorca klasy vector 29
1.12. Wyświetlanie komunikatów o błędach i implementacja wskaźników postępu 29

Rozdział 2. Algebra macierzy i równania liniowe 33

2.1. Metoda bezpośredniego rozwiązywania układu równań macierzowych metodą eliminacji Gaussa 34
2.1.1. Skalowanie układu równań liniowych 38
2.2. Rozwiązywanie układu równań liniowych według algorytmu Crouta 40
2.3. Obliczanie macierzy odwrotnej metodą eliminacji Gaussa 44
2.4. Obliczanie macierzy odwrotnej metodą Crouta 48
2.5. Obliczanie wyznacznika macierzy kwadratowej 53
2.6. Wskaźnik uwarunkowania macierzy 54
2.7. Obliczanie wartości własnej macierzy kwadratowej A o największym module 56
2.8. Obliczanie wartości własnej macierzy $1 - \alpha A$ o największym module 57
2.9. Rozwiązywanie układu równań liniowych metodą iteracji Jacobiego oraz Richardsona 59
2.10. Rozwiązywanie układu równań metodą Gaussa-Seidela oraz metodą nadrelaksacji 62
2.11. Pseudorozwiązanie układu nadokreślonego 65
2.12. Metoda najmniejszych kwadratów 71
2.13. Algorytm Crouta rozwiązywania rzadkich układów równań liniowych 73
2.14. Algorytmy iteracyjne Richardsona oraz Gaussa-Seidela dla macierzy rzadkich 82
Przykłady 88

Rozdział 3. Praktyka badania funkcji ... 111
3.1. Całkowanie i różniczkowanie numeryczne ... 111
3.1.1. Ekstrapolacja iterowana Richardsona i Aitkena ... 111
3.1.2. Całkowanie numeryczne ... 119
3.1.3. Różniczkowanie numeryczne ... 131
3.1.4. Gradient funkcji wielu zmiennych ... 142
3.1.5. Jacobian funkcji wektorowej wielu zmiennych ... 145
3.1.6. Hesjan funkcji wielu zmiennych ... 147
3.2. Wybrane metody aproksymacji i interpolacji liniowej funkcji jednej zmiennej ... 149
3.2.1. Aproksymacja metodą najmniejszych kwadratów ... 150
3.2.2. Aproksymacja funkcji dyskretnej wielomianem ... 152
3.2.3. Aproksymacja układami funkcji ortogonalnych ... 153
3.2.4. Aproksymacja wielomianami ortogonalnymi ... 154
3.2.5. Implementacja metod aproksymacji ... 156
3.2.6. Interpolacja funkcji dyskretnej krzywą łamaną ... 169
3.2.7. Interpolacja wielomianem potęgowym Lagrange'a ... 170
3.2.8. Interpolacja funkcjami sklejanymi ... 170
3.2.9. Interpolacja funkcjami i wielomianami ortogonalnymi ... 172
3.2.10. Metody interpolacji w ramach klasy TInterpolacja ... 175
3.3. Wybrane metody poszukiwania minimum funkcji wielu zmiennych metodami bezgradientowymi ... 189
3.3.1. Wyznaczenie minimum funkcji wielu zmiennych bezgradientową metodą poszukiwań prostych Hooke'a-Jeevesa ... 191
3.3.2. Bezgradientowa metoda „złotego podziału" poszukiwania minimum ... 193
3.3.3. Bezgradientowa metoda Powella poszukiwania minimum funkcji wielu zmiennych ... 201
3.4. Wybrane metody poszukiwania minimum funkcji wielu zmiennych metodami gradientowymi ... 205
3.4.1. Metoda ekspansji i kontrakcji geometrycznej z jednym testem badania współczynnika kroku przy poszukiwaniu minimum w kierunku ... 206
3.4.2. Metoda aproksymacji parabolicznej z jednym testem badania współczynnika kroku przy poszukiwaniu minimum w kierunku ... 210
3.4.3. Algorytm największego spadku ... 214
3.4.4. Zmodyfikowany algorytm Newtona ... 217
Przykłady ... 222

Rozdział 4. Równania nieliniowe, zera wielomianów, wartości własne macierzy . 263
4.1. Algorytmy rozwiązywania układów równań nieliniowych ... 264
4.1.1. Rozwiązywanie układów równań nieliniowych metodą Newtona ... 265
4.1.2. Rozwiązywanie układów równań nieliniowych metodą gradientową ... 268
4.1.3. Rozwiązywanie układu równań nieliniowych zmodyfikowaną metodą Newtona ... 271
4.1.4. Rozwiązywanie układów nieliniowych metodą iteracyjną ... 275
4.1.5. Pseudorozwiązania nieliniowego układu nadokreślonego metodą Hooke'a-Jeevsa ... 278
4.2. Wyznaczanie zer wielomianów metodami Bairstowa i Laguerre'a ... 280
4.2.1. Dzielenie wielomianów o współczynnikach rzeczywistych przez czynnik liniowy według algorytmu Hornera ... 280
4.2.2. Dzielenie wielomianu przez czynnik kwadratowy ... 282
4.2.3. Wyznaczanie dzielników wielomianu stopnia N>2 w postaci trójmianu kwadratowego metodą Bairstowa ... 282
4.2.4. Wyznaczanie zer wielomianów o współczynnikach rzeczywistych ... 287
4.2.5. Wyznaczanie zera wielomianu metodą Laguerre'a ... 288
4.2.6. Wyznaczanie wszystkich zer wielomianu metodą Laguerre'a ... 290

4.3. Wyznaczanie wartości własnych macierzy metodami Bairstowa i Laguerre'a.....293
4.3.1. Wyznaczanie współczynników wielomianu charakterystycznego macierzy kwadratowej metodą Kryłowa.....293
4.3.2. Wyznaczanie wartości własnych macierzy metodą Bairstowa.....295
4.3.3. Wyznaczanie wartości własnych macierzy metodą Laguerre'a.....297
4.4. Wyznaczanie zer funkcji jednej zmiennej metodą połowienia przedziału.....298
Przykłady.....299

Rozdział 5. Układy zwyczajnych równań różniczkowych nieliniowych 315
5.1. Układ równań różniczkowych jako klasa programowania obiektowego 317
5.1.1. Definicje typów do zadawania układu równań różniczkowych nieliniowych.....317
5.1.2. Definicja klasy prototypowej dla klas implementujących rozwiązywanie układu równań różniczkowych 318
5.1.3. Definicja klasy prototypowej dla klas potomnych dotyczących rozwiązywania układu równań różniczkowych nieliniowych 324
5.1.4. Aproksymacja dyskretnych wartości wektorów stanu.....327
5.1.5. Funkcje pomocnicze do działania na wektorach stanu 330
5.2. Metody Rungego-Kutty.....331
5.3. Rozwiązywanie układu równań różniczkowych zwyczajnych metodą Rungego-Kutty z automatycznym doborem kroku całkowania.....337
5.4. Metody Fehlberga.....341
5.5. Rozwiązanie układu równań różniczkowych nieliniowych zwyczajnych metodą Fehlberga z automatycznym doborem kroku całkowania 349
5.6. Rozwiązanie układu równań różniczkowych nieliniowych zwyczajnych metodą Dormanda-Prince'a z automatycznym doborem kroku całkowania 352
5.7. Metoda wielokrokowa rozwiązywania układu równań różniczkowych nieliniowych z członem przewidywania Adamsa-Bashfortha oraz członem korekcyjnym Adamsa-Multona z automatycznym doborem kroku i rzędu.....358
5.7.1. Algorytm Adamsa-Bashfortha 358
5.7.2. Algorytm Adamsa-Multona 360
5.7.3. Algorytmy przewidywania i korekcji wyrażone przez macierz Nordsiecka 363
5.7.4. Faza wstępna obliczeń.....373
5.7.5. Metody klasy TAdamsMultonAbstract i TAdamsMulton realizujące algorytm Adamsa-Multona 377
5.8. Rozwiązywanie układu równań nieliniowych metodą sztywno stabilnych algorytmów Geara.....383
5.9. Metoda Gragga z ekstrapolacją Bulirscha-Stoera.....395
Przykłady.....403

Rozdział 6. Układy równań różniczkowych liniowych o stałych współczynnikach .. 425
6.1. Równania różnicowe dla różnych aproksymacji funkcji wymuszających.....429
6.1.1. Wymuszenie aproksymowane funkcjami przedziałami stałymi 430
6.1.2. Wymuszenie aproksymowane funkcjami przedziałami liniowymi 432
6.1.3. Wymuszenie aproksymowane wielomianem stopnia drugiego 434
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 436
6.2. Definicja typów dla liniowych równań różniczkowych 438
6.3. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami stałymi.....441

6.4. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami liniowymi ... 444
6.5. Numeryczne rozwiązywanie równań różniczkowych liniowych o stałych współczynnikach dla aproksymacji wymuszeń funkcjami przedziałami kwadratowymi ... 447
Przykłady ... 450

Rozdział 7. Praktyka przekształceń Fouriera ... 457
7.1. Dyskretna transformacja Fouriera według algorytmu Hornera ... 463
7.2. Szybkie przekształcenie Fouriera według algorytmu Cooleya-Tukeya ... 465
7.3. Szybkie przekształcenie Fouriera według algorytmu Sandego-Tukeya ... 473
7.4. Wyznaczanie współczynników zespolonego szeregu Fouriera dla dowolnej funkcji okresowej ... 477
7.5. Obliczanie odwrotnej transformacji Fouriera dla dowolnej transformaty ... 478
Przykłady ... 480

Rozdział 8. Praktyka przekształceń Laplace'a ... 495
8.1. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasu z zastosowaniem szeregów Fouriera ... 496
8.2. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasowej z zastosowaniem szeregów Laguerre'a ... 502
8.3. Numeryczne obliczanie transformacji odwrotnej Laplace'a w wybranej chwili czasowej według algorytmu Valsa ... 506
8.4. Obliczanie transformacji odwrotnej Laplace'a funkcji wymiernej na podstawie jej pozostałości w biegunach ... 510
8.4.1. Definiowanie klasy do obliczania odwrotnej transformacji Laplace'a funkcji wymiernej na podstawie jej pozostałości w biegunach ... 513
Przykłady ... 518

Bibliografia ... 531

Skorowidz ... 535

Rozdział 6.

Układy równań różniczkowych liniowych o stałych współczynnikach

Zadany jest układ N równań różniczkowych liniowych niejednorodnych:

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

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

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{6.2}$$

gdzie:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \cdot \\ \cdot \\ \cdot \\ x_N(t) \end{bmatrix}; \quad \frac{d\mathbf{x}(t)}{dt} = \begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \\ \cdot \\ \cdot \\ \cdot \\ \frac{dx_N(t)}{dt} \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} a_{11}a_{12}\ldots a_{1N} \\ a_{21}a_{22}\ldots a_{2N} \\ \ldots\ldots \\ a_{N1}a_{N2}\ldots a_{NN} \end{bmatrix};$$

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

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

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

Szereg macierzowy (6.3) jest równoważny z N^2 zwykłych skalarnych szeregów potęgowych:

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

Dla 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)

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

2. Jeżeli macierz **A** komutuje z macierzą **B**, a więc **AB** = **BA**, to:

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

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:

$$\left[e^{\mathbf{A}t}\right]^{-1} = e^{-\mathbf{A}t}. \tag{6.6}$$

4. Różniczkując obie strony równania macierzowego (6.3) ze względu na t oraz wyłączając wspólny czynnik **A** z wyrazów szeregu nieskończonego, otrzymujemy:

$$\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A}. \tag{6.7}$$

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

$$\int_{t_1}^{t_2} e^{\mathbf{A}t}\,dt = \mathbf{A}^{-1}\left(e^{\mathbf{A}t_2} - e^{\mathbf{A}t_1}\right) = \left(e^{\mathbf{A}t_2} - e^{\mathbf{A}t_1}\right)\mathbf{A}^{-1}. \tag{6.8}$$

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

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

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

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{y}, \tag{6.10}$$

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

Istotnie, z własności (6.7) wynika

$$\frac{dx(t)}{dt} = \frac{d}{dt}\left(e^{\mathbf{A}t}y(\mathbf{t})\right) = \mathbf{A}e^{\mathbf{A}t}y = \mathbf{A}x(t). \tag{6.11}$$

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

$$\mathbf{x}(t) = e^{\mathbf{A}t}\,\mathbf{y}(t), \tag{6.12}$$

a następnie podstawia się wyrażenie (6.12) do równania niejednorodnego (6.2) z uwzględnieniem własności (6.7)

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

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

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

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

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

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

$$\mathbf{y}(t_0) = e^{-\mathbf{A}t}\,\mathbf{x}(t_0). \tag{6.16}$$

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

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

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

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

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

Niech zachodzi przypadek ogólny, dla którego macierz **A** ma dzielniki elementarne:

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

gdzie wśród wartości własnych $\lambda_1, \lambda_2, \ldots, \lambda_s$ macierzy **A** będących, zgodnie z definicją, zerami wielomianu charakterystycznego macierzy **A**

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

mogą być liczby jednakowe; $1 \le p_n \le N$, przy czym $p_1 + p_2 + \ldots + p_s = M$. Dowodzi się, że w takim przypadku istnieje taka macierz nieosobliwa **S**, że

$$\mathbf{A} = \mathbf{S}^{-1}\,\mathbf{C}\mathbf{S}, \tag{6.19}$$

gdzie macierz **C** jest macierzą quasi-diagonalną zwaną kanoniczną macierzą Jordana [30]

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_{p1}(\lambda_1) & 0 & \ldots & 0 \\ 0 & \mathbf{I}_{p1}(\lambda_2) & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ 0 & 0 & \ldots & \mathbf{I}_{pi}(\lambda_i) \end{bmatrix}; \ \mathbf{I}_{pi}(\lambda_i) = \begin{bmatrix} \lambda_i & 0 & 0 & \ldots & 0 & 0 \\ 1 & \lambda_i & 0 & \ldots & 0 & 0 \\ 0 & 1 & \lambda_i & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & \lambda_i & 0 \\ 0 & 0 & 0 & \ldots & 1 & \lambda_i \end{bmatrix} \tag{6.20}$$

Stosując transformację (6.19), funkcję wykładniczą $e^{\mathbf{A}t}$ możemy przekształcić następująco:

$$e^{\mathbf{A}t} = e^{(\mathbf{S}^{-1}\mathbf{C}\mathbf{S})t} = e^{\mathbf{S}^{-1}(\mathbf{C}t)\mathbf{S}} = \mathbf{S}^{-1}e^{\mathbf{C}t}\mathbf{S} \ . \tag{6.21}$$

Ponieważ macierz **C** jest quasi-diagonalną, to:

$$e^{\mathbf{C}t} = \begin{bmatrix} e^{\mathbf{I}_{p1}(\lambda_1)t} & 0 & 0 \\ 0 & e^{\mathbf{I}_{p2}(\lambda_2)t} & 0 \\ . & . & . \\ 0 & 0 & e^{\mathbf{I}_{ps}(\lambda_s)t} \end{bmatrix}. \tag{6.22}$$

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

$$e^{\mathbf{I}_{p1}(\lambda_i)t} = \begin{bmatrix} e^{\lambda_i t} & 0 & 0 & \dots & 0 \\ te^{\lambda_i t} & e^{\lambda_i t} & 0 & \dots & 0 \\ \frac{t^2}{2!}e^{\lambda_i t} & te^{\lambda_i t} & e^{\lambda_i t} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \frac{t^{p_i-1}}{(p_i-1)!}e^{\lambda_i t} & \frac{t^{p_i-2}}{(p_i-2)!}e^{\lambda_i t} & \frac{t^{p_i-3}}{(p_i-3)!}e^{\lambda_i t} & \dots & e^{\lambda_i t} \end{bmatrix}. \tag{6.23}$$

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 $P_i(t)$ nie większego stopnia 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

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

Jeżeli $\mathrm{Re}\{\lambda_i\} = \alpha_i > 0$, 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 $\mathrm{Im}\{\lambda_i\} = \beta_i \neq 0$, 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 $P_i(t)$

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

6.1. Równania różnicowe dla różnych aproksymacji funkcji wymuszających

Do numerycznego rozwiązania układu równań różniczkowych liniowych (6.2) możemy 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:

$$\mathbf{u}(t) = \mathbf{u}(kT) \text{ dla } kT \le t \le (k+1)T,\ k = 0,1,2,\ldots \tag{6.26}$$

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

$$\int_{\mathrm{kT}}^{(\mathrm{k+1})\mathrm{T}} \mathrm{e}^{-\mathbf{A}\tau}\mathbf{Bu}(\tau)\mathrm{d}\tau = -\mathrm{e}^{-\mathbf{A}\tau}\Big|_{\mathrm{kT}}^{(\mathrm{k+1})\mathrm{T}} \mathbf{A}^{-1}\mathbf{Bu}(\mathrm{kT}) = \tag{6.27}$$
$$= \left(-\mathrm{e}^{-\mathbf{A}(\mathrm{k+1})\mathrm{T}} + \mathrm{e}^{-\mathbf{A}k\mathrm{T}}\right)\mathbf{A}^{-1}\mathbf{Bu}(\mathrm{kT}).$$

Umieszczając powyższy wynik całkowania w równaniu (6.18), otrzymujemy:

$$\mathbf{x}\left[(\mathrm{k}+1)\mathrm{T}\right] = \mathrm{e}^{\mathbf{A}\mathrm{T}}\mathbf{x}(\mathrm{kT}) + \mathrm{e}^{\mathbf{A}(\mathrm{k+1})\mathrm{T}}\left(-\mathrm{e}^{-\mathbf{A}(\mathrm{k+1})\mathrm{T}} + \mathrm{e}^{-\mathbf{A}\mathrm{kT}}\right)\mathbf{A}^{-1}\mathbf{Bu}(\mathrm{kT}) = \tag{6.28}$$
$$= \mathrm{e}^{\mathbf{A}\mathrm{T}}\mathbf{x}(\mathrm{kT}) + \left(\mathrm{e}^{\mathbf{A}\mathrm{T}} - \mathbf{1}\right)\mathbf{A}^{-1}\mathbf{Bu}(\mathrm{kT}),$$

gdzie: **1** — macierz jednostkowa.

W równaniu różnicowym (6.28) powinniśmy, 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:

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

wnikającą z definicji (6.3).

Gdy uwzględnimy więc równanie (6.29) oraz oznaczenia macierzy:

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

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

i wektorów

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

formuła rekurencyjna (6.28) przyjmie postać:

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

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, powinniśmy zauważyć, że formowanie macierzy $\mathbf{F}$ i $\mathbf{G}_0$ (wzory (6.30) i (6.31)) należy prowadzić równolegle 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, musimy 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:

```
template <class Typ>
int    MacPom1(vector<vector<Typ> > &A,vector<vector<Typ> > &B,
                     vector<vector<Typ> > &F,vector<vector<Typ> > &G1,
                    Typ T, Typ eps, Typ epsw , int N, int W)
{
  int K,Blad ;
  Typ  S,S1,NormaAT,teta,MWA ;
  vector<vector<Typ> > AX,AY,AT,BX,BT  ;
  InicjacjaMacierzy(AT,N,N);   InicjacjaMacierzy(AX,N,N);
  InicjacjaMacierzy(BX,N,N);
  InicjacjaMacierzy(AY,N,N);     InicjacjaMacierzy(BT,N,W);
  try
    {
      AT=A*T;
      MacierzJednostkowa(AX)  ;
      MacierzJednostkowa(F)  ;   MacierzJednostkowa(BX)  ;
      NormaAT=NormaMacierzy(AT);
      K=0; S=1; S1=1;
      teta=NormaAT/(1-NormaAT);
      do
      {
          K++;
          AY=AX*AT  ;
          S/=K;  AX=AY*S; F=F+AX   ;
          S1/=K+1;  AX=AY*S1 ;
          BX=BX+AX ;
          AX=Kopiuj(AY);
          teta*=NormaAT/(K+1);
      } while (teta>=eps);
      MWA=MWWMacA(F,epsw,800) ;
      if (MWA>=1.05) return 16;
      BT=B*T;  G1=BX*BT   ;
      return 0;
    }
  catch(...)
   {
    PiszKomBladRozRozn(17);
    return 17;
   }
}
```

6.1.2. Wymuszenie aproksymowane funkcjami przedziałami liniowymi

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

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

dla

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

gdzie:

$$\mathbf{C}_{i} = \frac{2^{i}}{T^{i+1}} \int_{0}^{T} e^{\mathbf{A}(T-\tau)} s^{i}\, ds \quad ; \quad \mathbf{f}_2 = \frac{1}{T}\left[\mathbf{u}\left((k+1)T\right) - \mathbf{u}(kT)\right]. \tag{6.34a}$$

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

$$\int_{kT}^{(k+1)T} e^{\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau = -\mathbf{A}^{-1}e^{-\mathbf{A}\tau}\mathbf{B}\left(\mathbf{f}_1 + \mathbf{f}_2\tau\right)\Big|_{kT}^{(k+1)T} + \int_{kT}^{(k+1)T}\mathbf{A}^{-1}e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{f}_2 d\tau =$$

$$= \mathbf{A}^{-1}e^{-\mathbf{A}kT}\left\{\left[\mathbf{B} + \mathbf{A}^{-1}\left(e^{-\mathbf{A}kT} - \mathbf{1}\right)\mathbf{B}\frac{1}{T}\right]\mathbf{u}(kT) + \right.$$

$$\left. + \left[e^{-\mathbf{A}T}\mathbf{B} - \mathbf{A}^{-1}\left(e^{-\mathbf{A}T} - \mathbf{1}\right)\mathbf{B}\frac{1}{T}\right]\mathbf{u}(k+1)T\right\}.$$

Gdy uwzględnimy powyższy wynik całkowania oraz oznaczenie (6.32), równanie różnicowe (6.18) przyjmie postać:

$$\mathbf{x}(k+1) = e^{\mathbf{A}T}\mathbf{x}(\mathbf{k}) + \mathbf{A}^{-1}\left[e^{\mathbf{A}T} - \left(e^{\mathbf{A}T} - \mathbf{1}\right)\mathbf{A}^{-1}\frac{1}{T}\right]\mathbf{B}\mathbf{u}(k) + \tag{6.35}$$

$$+ \mathbf{A}^{-1}\left[\left(e^{\mathbf{A}T} - \mathbf{1}\right)\mathbf{A}^{-1}\frac{1}{T} - 1\right]\mathbf{B}\mathbf{u}(k+1).$$

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

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

gdzie:

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

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

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 z wzorami (6.30), (6.37) i (6.38) należy prowadzić równolegle ze względu na wspólne elementy $(\mathbf{A}T)^n$ 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:

```
//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
//epsw - błąd wyznaczenia największej co do modułu wartości
//       własnej macierzy F
template <class Typ>
int   MacPom2(vector<vector<Typ> > &A, vector<vector<Typ> > &B,vector<vector<Typ> >
&F,
                      vector<vector<Typ> > &G2, vector<vector<Typ> > &H ,
                     Typ T, Typ eps, Typ epsw ,int N, int W)
{
  int K,Blad ;
  Typ SS,S1,S2,NormaAT,teta,MWA,a=0.5 ;
  vector<vector<Typ> >  AX,AY,AT,AG,AH,BT;
  InicjacjaMacierzy(AT,N,N);   InicjacjaMacierzy(AX,N,N);
  InicjacjaMacierzy(AY,N,N);   InicjacjaMacierzy(AH,N,N);
  InicjacjaMacierzy(AG,N,N);   InicjacjaMacierzy(BT,N,W);
  try
  {
    AT=A*T;       MacierzJednostkowa(AX);
    NormaAT=NormaMacierzy(AT);
    K=0; SS=1; S1=0.5; S2=0.5;
    teta=NormaAT/(1-NormaAT);
    MacierzJednostkowa(F);
    AG=AX*a;  AH=AX*a;
    do
    {
          K++;
          AY=AX*AT ;
          SS/=K;
          AX=AY*SS; F=F+AX ;
          S1*=(K+1)/((K+2)*K);
          AX=AY*S1;  AG=AG+AX ;
          S2/=K+2;
          AX=AY*S2; AH=AH+AX;
          for (int i=0; i<=N; i++)
          for (int j=0; j<=N; j++) AX[i][j]=AY[i][j];
          teta*=NormaAT/(K+1) ;
     } while (teta>=eps);
     MWA=MWWMacA(F,epsw,800) ;
     if (MWA>=1.05)  return 16;
     BT=B*T;  G2=AG*BT;  H=AH*BT;
     return 0;
    }
  catch(...)
```

```
    {
      PiszKomBladRozRozn(19);
      return 19;
    }
}
```

6.1.3. Wymuszenie aproksymowane wielomianem stopnia drugiego

W tym przypadku dokonujemy we wzorze (6.18) zamiany zmiennych pod całką:

$$\tau = s + kT \quad \text{gdzie } 0 < s < T,$$

co daje:

$$\mathbf{x}\left[(k+1)T\right] = e^{\mathbf{A}T}\mathbf{x}(kT) + \int_0^T e^{\mathbf{A}(T-s)}\mathbf{B}\mathbf{u}(kT+s)ds\ . \tag{6.39}$$

W celu obliczenia całki we wzorze (6.39) należy interpolować $\mathbf{u}(kT+s)$ wielomianem drugiego stopnia:

$$\mathbf{u}(\mathrm{kT+s}) = \left(1 - \frac{3\mathrm{s}}{\mathrm{T}} + \frac{2\mathrm{s}^2}{\mathrm{T}^2}\right)\mathbf{u}(\mathrm{kT}) + \left(\frac{4\mathrm{s}}{\mathrm{T}} - \frac{4\mathrm{s}^2}{\mathrm{T}^2}\right)\mathbf{u}\left(\mathrm{kT} + \frac{\mathrm{T}}{2}\right) + \tag{6.40}$$
$$+ \left(-\frac{\mathrm{s}}{\mathrm{T}} + \frac{2\mathrm{s}^2}{\mathrm{T}^2}\right)\mathbf{u}\left((\mathrm{k}+1)\mathrm{T}\right)$$

Podstawiając przybliżenie (6.40) pod całkę wzoru (6.39), otrzymujemy:

$$\int_0^{\mathrm{T}} e^{\mathbf{A}(\mathrm{T}-\mathrm{s})}\mathbf{B}\mathbf{u}(\mathrm{kT}+\mathrm{s})\mathrm{ds} = \left(\mathbf{C}_0 - \frac{3}{2}\mathbf{C}_1 + \frac{1}{2}\mathbf{C}_2\right)(\mathbf{B}\mathrm{T})\mathbf{u}(\mathrm{kT}) + \tag{6.41}$$
$$+ \left(2\mathbf{C}_1 - \mathbf{C}_2\right)(\mathbf{B}\mathrm{T})\mathbf{u}\left(\mathrm{kT} + \frac{\mathrm{T}}{2}\right) + \left(-\frac{1}{2}\mathbf{C}_1 + \frac{1}{2}\mathbf{C}_2\right)(\mathbf{B}\mathrm{T})\mathbf{u}\left((\mathrm{k}+1)\mathrm{T}\right).$$

gdzie:

$$\mathbf{C}_i = \frac{2^i}{T^{i+1}}\int_0^T e^{\mathbf{A}(T-s)}s^i ds \quad \text{dla } i = 0,1,2. \tag{6.42}$$

Obliczając całkę (6.42) dla poszczególnych $i = 0, 1, 2$, otrzymujemy:

$$\mathbf{C}_0 = \left(e^{\mathbf{A}T} - \mathbf{1}\right)(\mathbf{A}T) = \sum_{n=0}^{\infty}\frac{(\mathbf{A}T)^n}{(n+1)!},$$

$$\mathbf{C}_1 = 2\left(e^{\mathbf{A}T} - \mathbf{1} - \mathbf{A}T\right)\left[(\mathbf{A}T)^{-1}\right]^2 = 2\sum_{n=0}^{\infty}\frac{(\mathbf{A}T)^n}{(n+2)!}, \tag{6.43}$$

$$\mathbf{C}_2 = 8\left[e^{\mathbf{A}T} - \mathbf{1} - \mathbf{A}T - \frac{1}{2}(\mathbf{A}T)^2\right]\left[(\mathbf{A}T)^{-1}\right]^3 = 8\sum_{n=0}^{\infty}\frac{(\mathbf{A}T)^n}{(n+3)!}.$$

Gdy przyjmiemy oznaczenie (6.32) oraz uwzględnimy wynik całkowania (6.41), równanie rekurencyjne (6.39) przyjmuje postać:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}_2\mathbf{u}(k) + \mathbf{H}_2\mathbf{u}\left(k+\frac{1}{2}\right) + \mathbf{R}\mathbf{u}(k+1), \tag{6.44}$$

gdzie:

$$\mathbf{G}_2 = \left(\mathbf{C}_0 - \frac{3}{2}\mathbf{C}_1 + \frac{1}{2}\mathbf{C}_2\right)(\mathbf{B}T), \tag{6.45}$$

$$\mathbf{H}_2 = (2\mathbf{C}_1 - \mathbf{C}_2)(\mathbf{B}T), \tag{6.46}$$

$$\mathbf{R} = \left(-\frac{1}{2}\mathbf{C}_1 + \frac{1}{2}\mathbf{C}_2\right)(\mathbf{B}T), \tag{6.47}$$

natomiast macierz **F** wyraża się wzorem (6.30).

Równanie rekurencyjne (6.44) daje algorytm wyznaczania rozwiązania równania różniczkowego (6.2) w postaci dyskretnej $\mathbf{x}(k)$ (k = 0, 1,…). W celu zminimalizowania operacji numerycznych należy zauważyć, że formowanie macierzy **F**, $\mathbf{G}_2$, $\mathbf{H}_2$, **R** należy prowadzić równolegle (wzory (6.30), (6.43), (6.45), (6.46), (6.47)) ze względu na wspólne elementy $(\mathbf{A}T)^n$ występujące w szeregach macierzowych tych wzorów. Zauważmy również, że jeżeli $\mathbf{x}(t)$ jest liczone co T, to $\mathbf{u}(t)$ musi być zadane co $\frac{1}{2}T$ sekund.

Stosując wzór rekurencyjny (6.44) odpowiadający aproksymacji wymuszeń wielomianem stopnia drugiego, musimy w pierwszej kolejności wygenerować macierze **F**, $\mathbf{G}_2$, $\mathbf{H}_2$, **R** (wzory (6.30), (6.45), (6.46) i (6.47)). Można tego dokonać w sposób pokazany w poniższym bloku funkcyjnym:

```
//N    - rząd macierzy A i F
//W    - liczba kolumn macierzy B,G,H,R
//T    - wybrany krok całkowania
//eps  - górna granica błędu przybliżenia macierzy F,G,H,R
//epsw - błąd wyznaczenia największej co do modułu
//       wartości własnej macierzy F
template <class Typ>
int   MacPom3(vector<vector<Typ> > &A, vector<vector<Typ> > &B, vector<vector<Typ>
> &F,
                      vector<vector<Typ> > &G, vector<vector<Typ> > &H,
                      vector<vector<Typ> > &R , Typ T, Typ eps, Typ epsw ,
                      int N, int W)
{
  int K,Blad ;
  Typ SS,S1,S2,S3,NormaAT,teta,MWA,a1 ;
  vector<vector<Typ> > AX,AY,AT,AG,AH,AR,BT ;
  InicjacjaMacierzy(AT,N,N);   InicjacjaMacierzy(AX,N,N);
  InicjacjaMacierzy(AY,N,N);   InicjacjaMacierzy(AH,N,N);
  InicjacjaMacierzy(AG,N,N);   InicjacjaMacierzy(AR,N,N);
  InicjacjaMacierzy(BT,N,W);
  try
  {
```

```
    AT=A*T;       MacierzJednostkowa(AX);
    NormaAT=NormaMacierzy(AT);
    K=0; SS=1; S1=1; S2=0.5; S3=1.0/6.0;
    teta=NormaAT/(1-NormaAT);
    MacierzJednostkowa(F);  AG=AX*S1;    AH=AX*S2;  AR=AX*S3;
    do
    {
          K++;
          AY=AX*AT;   SS/=K;
          AX=AY*SS;  F=F+AX;   S1=SS/(K+1);
          AX=AY*S1;    AG=AG+AX;  S2=S1/(K+2);
          AX=AY*S2;  AH=AH+AX;    S3=S2/(K+3);
          AX=AY*S3;  AR=AR+AX;
          for (int i=0; i<=N; i++)
          for (int j=0; j<=N; j++) AX[i][j]=AY[i][j];
          teta*=NormaAT/(K+1) ;
    }
    while (teta>=eps);
    MWA=MWWMacA(F,epsw,800) ;
    if (MWA>=1.05)  return 16;
    a1=-3.0;  AX=AH*a1;
    a1=4.0;   AY=AR*a1;
    AG=AG+AX;  AG=AG+AY;
    a1=4.0;   AX=AH*a1;
    a1=-8.0;  AY=AR*a1;
    for (int i=0; i<=N; i++)
      for (int j=0; j<=N; j++) AT[i][j]=AH[i][j];
    AH=AX+AY;
    a1=-1.0;  AX=AT*a1;
    a1=4.0;   AY=AR*a1;
    AR=AX+AY;              BT=B*T;
    G=AG*BT;   H=AH*BT;  R=AR*BT;
    return 0;
  }
  catch(...)
   {
     PiszKomBladRozRozn(21);
     return 21;
   }
}
```

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

W punktach 6.1.1, 6.1.2, 6.1.3 podano metody zastępowania równania stanu (6.2) przybliżonymi równaniami różnicowymi. W równaniach tych pojawiają się macierze dane w postaci szeregów:

$$\mathbf{S} = \sum_{n=0}^{\infty} \frac{(\mathbf{A}T)^n}{n! s_n} \quad \text{gdzie} \quad s_n \geq 1 \quad \text{dla} \quad n > 2 \tag{6.48}$$

(patrz wzory (6.30), (6.31), (6.37), (6.38), (6.45), (6.46), (6.47)). W obliczeniach przybliżamy **S**, biorąc skończoną liczbę składników rozwinięcia (6.48)

$$\mathbf{S} \cong \sum_{n=0}^{K} \frac{(\mathbf{A}T)^n}{n!s_n} = \mathbf{M} = \left[m_{ij}\right] \tag{6.49}$$

Macierz błędu wynosi wówczas

$$\mathbf{R} = \sum_{n=K+1}^{\infty} \frac{(\mathbf{A}T)^n}{n!s_n} = \left[r_{ij}\right] \tag{6.50}$$

Niech $\|\mathbf{A}\|$ oznacza jedną z następujących norm macierzy kwadratowej **A** stopnia *N*:

$$\|\mathbf{A}\| = \max_i \sum_{j=1}^{N} |a_{ij}| \tag{6.51}$$

lub

$$\|\mathbf{A}\| = \max_j \sum_{i=1}^{N} |a_{ij}| \tag{6.52}$$

Zgodnie z definicjami (6.51) lub (6.52) dla macierzy **R** zachodzi

$$|r_{ij}| \le \|\mathbf{R}\| \tag{6.53}$$

dla wszystkich *i*, *j*. Ponadto zachodzi [7]:

$$\|\mathbf{AB}\| \le \|\mathbf{A}\| \le \|\mathbf{B}\| . \tag{6.54}$$

Uwzględniając nierówności (6.53), (6.54) oraz równania (6.50), otrzymujemy (por. [7])

$$\begin{aligned} |r_{ij}| &\le \sum_{n=K+1}^{\infty} \frac{\mathbf{1}}{n!s_n} |\textit{element } (i, j) \textit{ w macierzy } (\mathbf{A}T)^n| \le \\ &\le \sum_{n=K+1}^{\infty} \frac{\mathbf{1}}{n!s_n} \|\mathbf{A}T\|^n \le \frac{\|\mathbf{A}T\|^{K+1}}{(K+1)!S_{K+1}} \left[1 + \|\mathbf{A}T\| + \|\mathbf{A}T\|^2 + \ldots\right]. \end{aligned} \tag{6.55}$$

Jeżeli krok całkowania *T* jest tak dobrany, że $\|\mathbf{A}T\| < 1$, to szereg w nierówności (6.55) jest zbieżny, co daje:

$$|r_{ij}| \le \frac{\|\mathbf{A}T\|^{K+1}}{(K+1)!S_{K+1}} \frac{1}{1-\|\mathbf{A}T\|} = \varepsilon . \tag{6.56}$$

Nierówność (6.56) daje górną granicę błędu elementów macierzy **M** (6.49) używanej jako przybliżenie macierzy **S** (6.48). Dla danego obcięcia **K** sumy szeregów typu (6.48) otrzymuje się największy błąd (6.56) dla przypadku gdy $s_n = 1$ ($n = 0, 1,\ldots$), co odpowiada macierzy $\mathbf{F} = e^{\mathbf{A}T}$ (wzór (6.30)). Ponieważ macierz ta występuje w konstrukcji wszystkich trzech rozpatrywanych algorytmów (6.33), (6.36), (6.44), jako kryterium błędu formowania macierzy występujących w tych algorytmach można przyjąć górny błąd obcięcia w postaci (por. [9]):

$$\varepsilon = \frac{\left\|\mathbf{A}T\right\|^{K+1}}{(K+1)!}\,\frac{1}{1-\left\|\mathbf{A}T\right\|}. \tag{6.57}$$

Przy zadanym ε wzór (6.57) może posłużyć do wyznaczania kroku całkowania T. Jak pokazano we wstępie, struktura rozwiązania równania (6.2) zależy od wartości własnych macierzy **A**. Wykazuje się, że w celu zagwarantowania stabilności numerycznej, krok czasowy całkowania T i liczba wyrazów $K + 1$ w rozwinięciu (6.49) muszą być tak dobrane, ażeby największa co do modułu wartość własna macierzy **M** (wzór (6.49)) była mniejsza od jedności. Warunki powyższe będą sprawdzane w procedurach obliczeniowych podanych niżej.