Janusz Grzędziński

ITERACYJNA METODA WYZNACZANIA PRĘDKOŚCI FLATTERU ZA POMOCĄ PEWNEJ FUNKCJI JEDNEJ ZMIENNEJ

WARSZAWA 1980

Praca wpłynęła do Redakcji dnia 5 stycznia 1980 r.

Zarejestrowana pod nr 10/1980

Na prawach rękopisu

Instytut Podstawowych Problemów Techniki PAN

Nakład 140 egz. Ark.wyd. 0,9. Ark.druk. 1,5.
Oddano do drukarni w marcu 1980 r.
Nr zamówienia 230/0/80

Warszawska Drukarnia Naukowa, Warszawa,
ul.Śniadeckich 8

Janusz Grzędziński
Zakład Mechaniki Cieczy i Gazów

ITERACYJNA METODA WYZNACZANIA PRĘDKOŚCI FLATTERU ZA POMOCĄ PEWNEJ FUNKCJI JEDNEJ ZMIENNEJ[x]

1. Wstęp

Zagadnienie automatycznego wyznaczania krytycznej prędkości flatteru polega na skonstruowaniu takiego algorytmu numerycznego rozwiązania równania flatteru, który spełniałby trzy podstawowe warunki:

1° Wynikiem działania algorytmu jest wartość prędkości lotu wyznaczona z zadaną dokładnością,

2° Znalezione rozwiązanie opisuje krytyczne warunki flatteru,

3° Nakład pracy obliczeniowej jest na tyle mały, że pozwala na wielokrotne wykorzystywanie algorytmu w zagadnieniach związanych z optymalizacją konstrukcji lotniczych.

Postawienie drugiego warunku wynika stąd, że równanie flatteru ma nieskończenie wiele rozwiązań dających w wyniku nieskończony zbiór możliwych prędkości, z których najniższa jest dopiero prędkością flatteru. Sformułowanie warunku trzeciego jest mniej precyzyjne, ponieważ zależy od dostępnego sprzętu obliczeniowego, a także od indywidualnego rachunku kosztów. Zawsze jednak czas i koszty obliczeń są na tyle duże, że ich zmniejszenie jest bardzo pożądane.

Wszystkie stosowane dotąd metody wyznaczania krytycznej prędkości flatteru bazują na klasycznym równaniu, w którym występują siły aerodynamiczne obliczone dla małych drgań harmonicznych konstrukcji. Dla tego równania nie ma teoretycznych metod

[x] Praca wykonana w ramach tematu 15.1 "Zagadnienia aeroelastyczności" problemu węzłowego 05.12 "Wytrzymałość i optymalizacja konstrukcji maszynowych i budowlanych".

prowadzących wprost do rozwiązania dającego najniższą spośród możliwych wartość prędkości. Trzeba zatem znaleźć wszystkie rozwiązania równania flatteru dające prędkości w zakresie prędkości eksploatacyjnych samolotu. Nie wiadomo jednak jaka jest liczba tych rozwiązań. Praktycznie jedynym, jak dotychczas, możliwym wyjściem jest przeszukanie całego zakresu prędkości eksploatacyjnych samolotu i to z dostatecznie małym krokiem, co z kolei zwiększa czas obliczeń. Warunki stawiane algorytmowi automatycznego wyznaczania prędkości flatteru są zatem ściśle ze sobą powiązane.

Wspomniany brak narzędzi teoretycznych sprawia, że przy istnieniu obecnie kilku metod automatycznego obliczania flatteru nie da się wyodrębnić jednej, która byłaby najbardziej użyteczna. Wymagania tego nie spełnia również metoda opisana w niniejszej pracy, ale wyróżnia się prostotą i dla układów aeroelastycznych o dużej liczbie stopni swobody często będzie szybciej zbieżna od innych metod, także wykorzystujących rozwiązanie flatterowego zagadnienia na wartości własne.

W pracy przedstawiono również tekst procedury realizującej algorytm obliczeń automatycznych (w języku ALGOL 1204).

2. Iteracyjne rozwiązywanie równania flatteru

Klasyczne równanie flatteru można zapisać następująco

$$\frac{b^2}{U^2}[K]\{u\} = k^2([M] + \varrho\,[A(k)])\{u\}, \tag{2.1}$$

gdzie:

U - prędkość lotu,

b - charakterystyczny wymiar liniowy,

ϱ - gęstość powietrza,

[K], [M] - macierze odpowiednio sztywności i mas układu,

[A(k)] - macierz aerodynamiczna (o elementach zespolonych),

$k = \omega b/U$ bezwymiarowy współczynnik częstości (ω - częstość kołowa drgań),

{u} - wektor opisujący postać flatteru.

W równaniu tym elementy macierzy aerodynamicznej [A(k)] mogą być obliczone tylko dla zadanych wartości współczynnika częstości k. Wielkościami niewiadomymi są współczynnik częstości k i prędkość lotu U. Aby równanie (2.1), które względem składowych

wektora {u} jest jednorodnym układem algebraicznych równań liniowych, miało rozwiązanie, musi być spełniony warunek

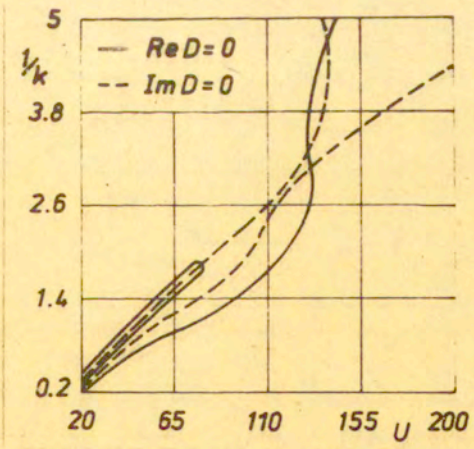
$$(2.2) \qquad D(k,U) \equiv \det\left(\frac{b^2}{U^2}[K] - k^2[M] - k^2 \varrho\, [A(k)]\right) = 0.$$

Ponieważ elementy macierzy aerodynamicznej są zespolonymi funkcjami współczynnika częstości, więc warunek (2.2) jest układem dwóch równań rzeczywistych

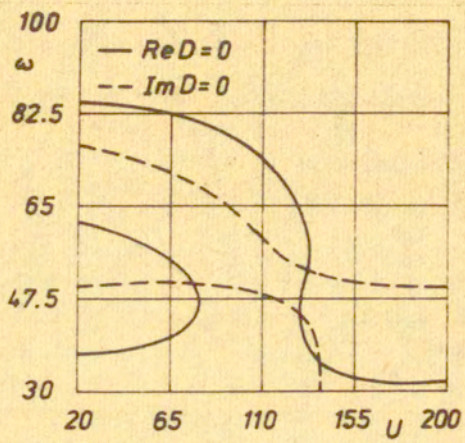
$$(2.3) \qquad \mathrm{Re}\, D(k,U) = 0, \qquad \mathrm{Im}\, D(k,U) = 0,$$

z których można wyznaczyć k i U. Są to jednak równania przestępne (ze względu na charakter zależności elementów macierzy aerodynamicznej od współczynnika częstości) spełnione przez nieskończenie wiele par k, U.

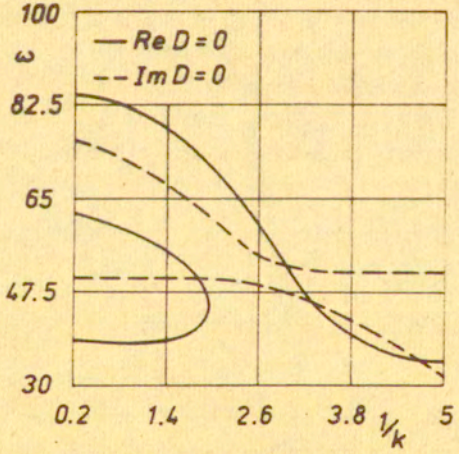
Układ równań (2.3) można rozwiązać w sposób przybliżony dowolną spośród znanych metod przybliżonego rozwiązywania układów równań nieliniowych. Jedną z częściej stosowanych metod jest reguła falsi [1,2] . Jako niewiadome można wybrać nie tylko k i U, ale również ω i U czy też ω i k. Każde z równań traktowane oddzielnie wyznacza na płaszczyźnie pewien zbiór krzywych. Punkty przecięcia krzywych obu tych zbiorów wyznaczają poszukiwane rozwiązania. Dla jednego z badanych układów aeroelastycznych otrzymano w ten sposób wykresy pokazane na rysunkach 1, 2 i 3, odpowiadających trzem możliwym parom niewiado-



Rys. 1



Rys. 2



Rys. 3

mych. Przed zastosowaniem metody iteracyjnej należy wybrać odpowiednio duży obszar zmienności obydwu niewiadomych i wstępnie zlokalizować miejsca zerowe wyznacznika. W tym celu dzieli się wybrany obszar na prostokąty dwoma układami linii równoległych i oblicza wartości części rzeczywistej i urojonej wyznacznika w węzłach powstałej w ten sposób siatki. Pierwiastki lokalizuje się na podstawie zmian znaku. Wymiary oczka siatki muszą być

dostatecznie małe, aby nie pominąć żadnego z rozwiązań. Optymalny wybór siatki zależy tu tylko od intuicji i doświadczenia osoby prowadzącej obliczenia i jest najsłabszym punktem metody. Na ogół do wstępnej lokalizacji trzeba obliczyć wartości wyznacznika w setkach punktów, a czasami i ta liczba może się okazać za mała. Z drugiej strony im dokładniej określi się położenie pierwiastka, tym mniej iteracji będzie trzeba do jego wyznaczenia. W efekcie opisana metoda niewiele różni się od zwykłego przeszukiwania wybranego obszaru z dostatecznie małym stałym krokiem.

Równanie (2.1) ma postać zagadnienia na wartości własne. Wykorzystanie tego faktu jest odmiennym od opisanego wyżej sposobem poszukiwania rozwiązań. Na podstawie równania (2.1) można sformułować kilka zagadnień na wartości własne, różniących się interpretacją samej wartości własnej. Dość często rozwiązuje się flatterowe zagadnienie na wartości własne w postaci

$$\lambda [K]\{u\} = ([M] + \varrho [A(k)])\{u\}, \tag{2.4}$$

przy czym w warunkach flatteru $\lambda = \omega^{-2}$ i jest liczbą rzeczywistą. Ponieważ macierz aerodynamiczna jest macierzą niehermitowską, więc dla założonej wartości współczynnika częstości k otrzyma się na ogół zbiór zespolonych wartości własnych. Poszukiwanie prędkości flatteru polega na znalezieniu takich wartości współczynnika częstości, dla których istnieją rzeczywiste wartości własne zagadnienia (2.4). Ta z nich, której odpowiada najniższa wartość prędkości, określa krytyczne warunki flatteru. Część urojona każdej wartości własnej jest nieliniową (przestępną) funkcją współczynnika częstości. Z punktu widzenia obliczeń numerycznych zagadnienie polega teraz na znalezieniu zer skończonej liczby nieliniowych funkcji jednej zmiennej. W porównaniu z poprzednim sposób ten ma istotne zalety. Po pierwsze wykorzystuje dobrze rozpracowane algorytmy numerycznego obliczania wartości własnych, a po drugie nie wymaga wstępnej lokalizacji zer. Poszukiwanie krytycznej prędkości flatteru w oparciu o wielokrotne rozwiązywanie zagadnienia na wartości własne jest obecnie najbardziej rozpowszechnione i stanowi podstawę kilku algorytmów obliczeń automatycznych [3,4,5,6].

Spośród problemów związanych z automatycznym wyznaczaniem

zer części urojonych wartości własnych (opisanych np. w pracy [7]) najpoważniejszym jest brak możliwości rozróżniania wartości własnych przy przejściu z jednej wartości współczynnika częstości do drugiej. Oznacza to, że nie można wybrać jednej wartości własnej i poszukiwać miejsc zerowych jej części urojonej, a następnie powtórzyć ten proces dla każdej kolejnej wartości własnej. W każdym kroku iteracyjnym można obliczyć n następnych przybliżeń (gdzie n jest liczbą wartości własnych), ale nie wiadomo, które z nich prowadzi do najbliższego miejsca zerowego. W związku z tym wybiera się następne przybliżenie leżące najbliżej poprzedniego, co w wielu przypadkach zmniejsza szybkość zbieżności procesu iteracyjnego.

3. Metoda jednej funkcji

Skoro zatem wartości własne zagadnienia flatterowego są nierozróżnialne w procesie iteracyjnym, to nie ma powodu, aby traktować je indywidualnie. Stwierdzenie to stanowi punkt wyjścia prezentowanej metody jednej funkcji. Metoda ta polega na skonstruowaniu takiej funkcji współczynnika częstości, która będzie miała zera w tych samych punktach (i tylko w tych punktach), w których występują zera części urojonych wszystkich wartości własnych. Mając taką funkcję zagadnienie wyznaczania prędkości flatteru sprowadzi się do wyznaczenia jej miejsc zerowych w zadanym przedziale zmienności współczynnika częstości przy pomocy jednej ze znanych metod przybliżonych dla funkcji nieliniowych jednej zmiennej.

Proponowana funkcja F(k) jest następująca

$$F(k) = s\left(\sum_{i=1}^{n} \frac{1}{|f_i(k)|}\right)^{-1}, \tag{3.1}$$

gdzie $f_i(k)$ oznacza część urojoną wartości własnej o numerze i ($i = 1,2,\ldots,n$), natomiast

$$s = \prod_{i=1}^{n} \operatorname{sign}(f_i).$$

Szczególna przydatność funkcji (3.1) do zastąpienia układu krzywych części urojonych wartości własnych wynika stąd, że o jej przebiegu decydują przede wszystkim te wartości własne, które

mają najmniejsze co do modułu części urojone. Z praktyki obliczeniowej wiadomo, że dla układów aeroelastycznych o dużej liczbie stopni swobody z reguły występuje pewna liczba wartości własnych, których części urojone nie mają zer, a dążą bardzo szybko do minus nieskończoności. Na zachowanie funkcji (3.1) mają one pomijalnie mały wpływ, podczas gdy w stosowanych obecnie algorytmach obliczeń automatycznych obliczone dla nich następne przybliżenia są zawsze brane pod uwagę. Fakt ten wskazuje również na mniejszą przydatność iloczynu

$$\prod_{i=1}^{n} f_i(k)$$

zamiast funkcji (3.1), ponieważ w iloczynie każdy czynnik jest niejako "równouprawniony".

Przy wyznaczaniu zer funkcji (3.1) wygodnie jest traktować ją jako funkcję odwrotności współczynnika częstości $F(x)$ $(x = k^{-1})$, a jako metodę iteracyjną wykorzystać zmodyfikowaną metodę Laguerre'a [4]. Wzór iteracyjny ma wówczas postać

$$x_{i+1} = x_i \pm \frac{F(x_i)}{\sqrt{(F'(x_i))^2 - F(x_i)F''(x_i)}} . \tag{3.2}$$

Występujące w nim pochodne są dane wzorami

$$\frac{dF}{dx} = sF^2 \sum_{i=1}^{n} \frac{\operatorname{sign}(f_i)}{f_i^2} \frac{df_i}{dx} ,$$

$$\frac{d^2F}{dx^2} = \frac{2}{F}\left(\frac{dF}{dx}\right)^2 + sF^2 \sum_{i=1}^{n} \frac{\operatorname{sign}(f_i)}{f_i^2}\left[\frac{d^2f_i}{dx^2} - \frac{2}{f_i}\left(\frac{df_i}{dx}\right)^2\right] . \tag{3.3}$$

Jak łatwo sprawdzić, w otoczeniu zera funkcji f_j $(1 \leqslant j \leqslant n)$ zachodzą związki

$$\lim_{f_j \to 0} F = 0, \qquad \lim_{f_j \to 0} \frac{dF}{dx} = \left.\frac{df_j}{dx}\right|_{f_j = 0} , \tag{3.4}$$

$$\lim_{f_j \to 0} \frac{d^2F}{dx^2} = \left.\frac{d^2 f_j}{dx^2}\right|_{f_j = 0},$$

czyli krotność każdego zera funkcji F(x) jest taka sama, jak krotność zera części urojonej odpowiedniej wartości własnej. W szczególności w otoczeniu punktu $x = 0$, gdzie $f_i = 0$ $(i = 1,2,\ldots,n)$

$$F(x) \approx s\left(\sum_{i=1}^{n} \frac{1}{|f_i'|}\right)^{-1} x + \cdots,$$

czyli jeżeli każda z funkcji $f_i(x)$ ma w punkcie $x = 0$ zero jednokrotne, to również funkcja F(x) ma w tym punkcie zero tego samego typu.

Przybliżenie początkowe x_0 można wybrać dowolnie. Natomiast jeżeli $F(x_0) = 0$, co ma miejsce po znalezieniu każdego kolejnego zera, to ze względu na wzór definicyjny (3.1) jest uzasadnione obliczenie następnego przybliżenia ze wzoru (3.2), w którym $x_i = x_0$, ale dla funkcji $F_j(x)$

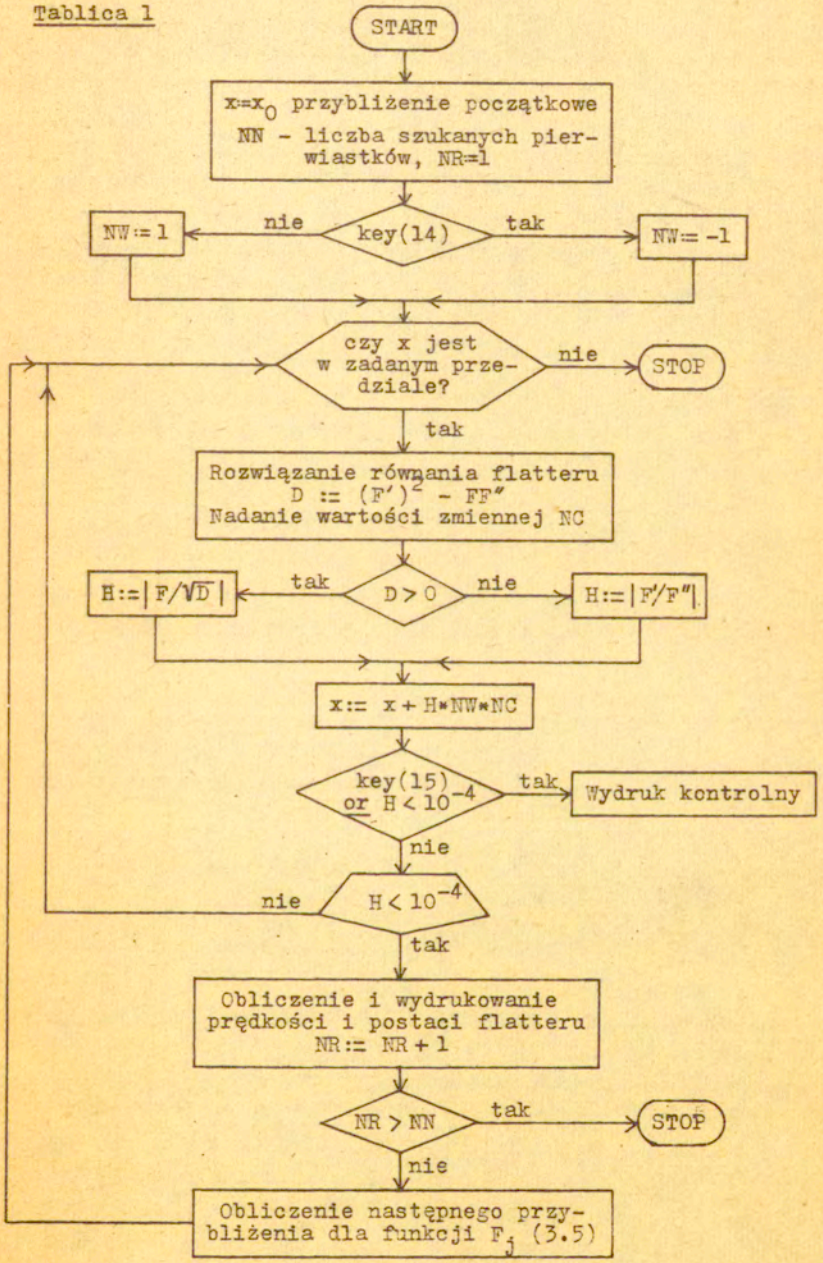
$$F_j(x) = s\left(\sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{1}{|f_i(x)|}\right)^{-1}, \tag{3.5}$$

gdzie j jest numerem wartości własnej, dla której $f_j(x_0) = 0$.

4. Algorytm wyznaczania prędkości flatteru

Algorytm automatycznego wyznaczania prędkości krytycznej flatteru przedstawiono w tablicy 1. Zadaniem algorytmu jest wyznaczenie zadanej liczby (lub wszystkich) miejsc zerowych w zadanym przedziale odwrotności współczynnika częstości. Każdej wartości x, dla której $F(x) = 0$, odpowiada pewna wartość prędkości lotu. Najniższa spośród znalezionych w ten sposób prędkości jest krytyczną prędkością flatteru. Zasadniczo prezentowany algorytm niewiele różni się od standardowego algorytmu rozwiązywania nieliniowego równania metodą iteracyjną. Oprócz wspomnianego już wymagania znalezienia zadanej liczby zer oraz zawiera-

Tablica 1



nia się argumentu w zadanym przedziale, wyróżniono kierunek schematu iteracyjnego. Otóż kolejne zera szukane są stale w kierunku rosnących (lub malejących) wartości argumentu aż do osiągnięcia zadanej ich liczby lub prawego (lewego) brzegu zadanego przedziału zmienności argumentu. Wprawdzie można jako przybliżenie początkowe wybrać lewy lub prawy koniec przedziału, ale liczba iteracji potrzebna do znalezienia leżących w nim zer będzie wtedy na ogół większa niż przy starcie z punktu leżącego wewnątrz (zwłaszcza, gdy jako przybliżenie początkowe zostanie wybrany lewy koniec przedziału leżący blisko zera). Stąd też w sieci działań przedstawionej w tablicy 1 pojawia się instrukcja

$$x := x + H*NW*NC,$$

gdzie H jest obliczonym ze wzoru (3.2) (lub ze wzoru Newtona dla pierwszej pochodnej) krokiem iteracji, natomiast zmienne NC i NW są zmiennymi sterującymi kierunkiem procesu iteracyjnego i mogą przyjmować wartości ± 1. Zmienna NW przyjmuje wartość $+1$ w przypadku dążenia w kierunku rosnących wartości argumentu i -1 w przypadku przeciwnym. Zmienna NC określa wybór znaku we wzorze (3.2). Jej wartość jest ustalana w każdym kroku w sposób następujący

$$NC_i := NC_{i-1}\,\mathrm{sign}(F_i F_{i-1}),$$

gdzie wskaźnik i numeruje iteracje. Jeżeli $|F_{i-1}| < 10^{-4}$ (zero znalezione z tolerancją 10^{-4}), to w powyższym wzorze zamiast F_{i-1} wstawiana jest wartość pierwszej pochodnej F'_{i-1}.

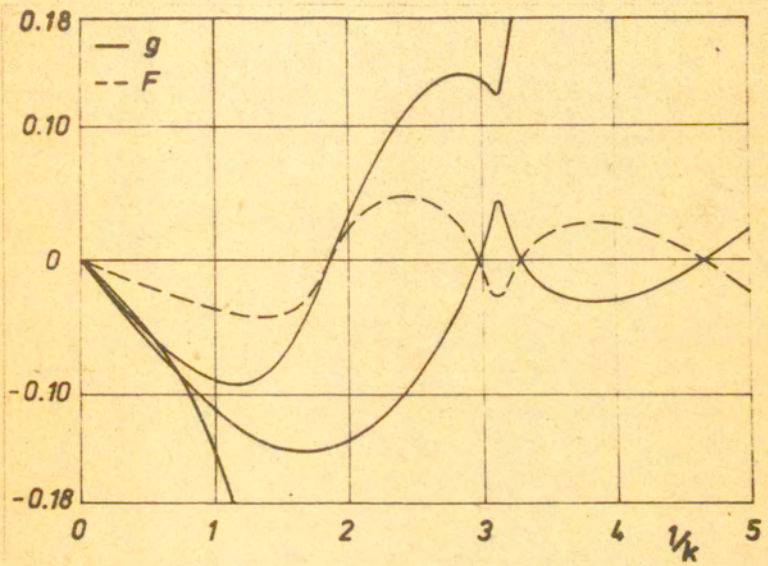
5. Przykład obliczeniowy

Obliczenia przykładowe wykonano dla układu aeroelastycznego o czterech stopniach swobody. Ponieważ układ miał jeden stopień swobody przemieszczenia sztywnego, więc jedna z wartości własnych była stale równa zeru. Do wyznaczenia prędkości flatteru wykorzystano zamiast części urojonych wartości własnych częściej używane krzywe tłumienia konstrukcyjnego g (mające zera w tych samych punktach, co części urojone wartości własnych), przedstawione na rysunku 4 wraz z wyznaczoną na ich podstawie funkcją F (3.1). Jest to ten sam układ aeroelastyczny, który był badany w pracy [7]. Dla funkcji F(x), gdzie jak poprzednio $x = k^{-1}$, obliczono następne przybliżenia według wzoru (3.2). Wy-

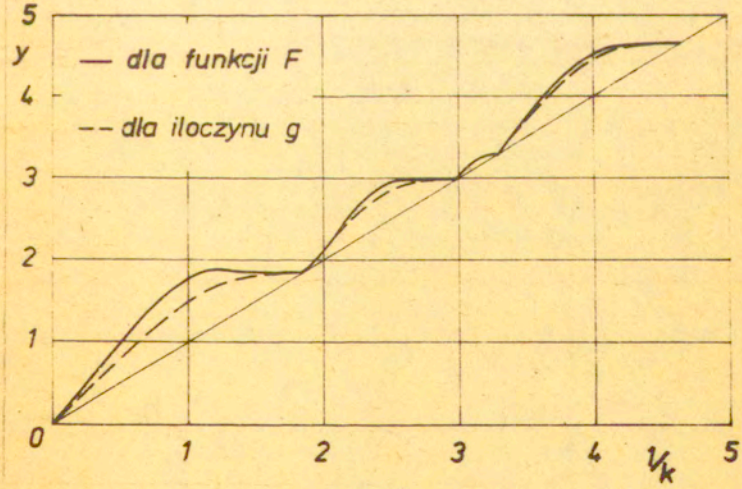
kres otrzymanej w ten sposób funkcji

(5.1) $$y = x + F/\sqrt{F'^2 - FF''}$$

przedstawiono na rysunku 5. Linią przerywaną pokazano na nim



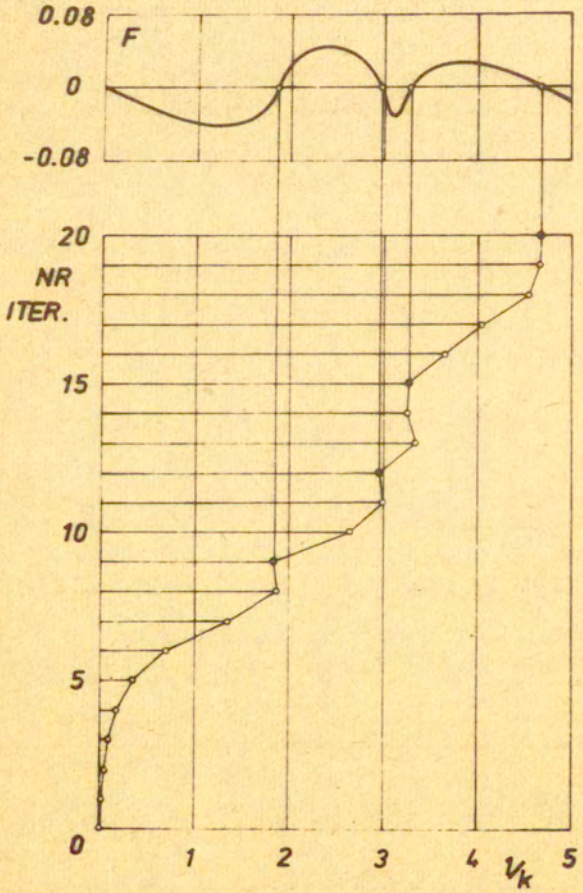
Rys. 4



Rys. 5

również wykres funkcji (5.1), ale obliczonej nie dla funkcji F, a dla iloczynu współczynników tłumienia odpowiadających poszczególnym wartościom własnym. Widać, że potwierdziły się przypuszczenia dotyczące wolniejszej zbieżności procesu iteracyjnego w tym przypadku.

Wyniki zastosowania opisanego w poprzednim rozdziale algorytmu iteracyjnego do badanego układu przedstawiono na rysunku 6 dla przybliżenia początkowego $1/k = 0.01$. Jako kryterium zna-



Rys. 6

lezienia pierwiastka przyjęto różnicę dwóch kolejnych wartości argumentu mniejszą od 10^{-4}. W przedstawionym przykładzie wybrano najbardziej niekorzystną wartość przybliżenia początkowego. Liczbę iteracji potrzebnych do znalezienia poszczególnych pierwiastków przy starcie z innego przybliżenia początkowego można oszacować bezpośrednio z wykresu przedstawionego na rysunku 5.

6. Użytkowanie procedury AUT5

Opisany algorytm iteracyjnego wyznaczania rozwiązań równania flatteru jest realizowany przy pomocy napisanej w języku ALGOL 1204 procedury o nazwie AUT5. Parametry formalne procedury są parametrami wejściowymi i w ich miejsce przy wywołaniu procedury trzeba podstawić wartości oznaczające kolejno:

FLAT - nazwa procedury rozwiązującej flatterowe zagadnienie na wartości własne,

b - charakterystyczny wymiar liniowy badanego układu aeroelastycznego,

Ml - całkowita liczba stopni swobody układu (wymiar macierzy aerodynamicznej),

N - różnica między całkowitą liczbą stopni swobody Ml i liczbą stopni swobody przemieszczeń sztywnych (liczba niezerowych wartości własnych oznaczona we wzorze (3.1) literą n),

Xl,XN - wartości argumentu 1/k będące odpowiednio dolną i górną granicą żądanego przedziału, w którym poszukiwane są zera funkcji F(1/k),

NN - liczba szukanych miejsc zerowych,

tlr - zadana dokładność wyznaczenia każdego zera,

XO - wartość przybliżenia początkowego.

Nagłówek procedury FLAT powinien mieć postać następującą

```
procedure FLAT(b,XO,Ml,N,WR,WI,RR,II,VR,VI,rA,iA);
value XO;
real XO,b;
integer Ml,N;
array WR,WI,RR,II,VR,VI,rA,iA;
```

Parametry formalne b, Ml, N są parametrami wejściowymi i mają to samo znaczenie, co parametry procedury AUT5 o tych samych

nazwach. Parametr wejściowy XO jest wartością 1/k, dla której rozwiązywane jest flatterowe zagadnienie na wartości własne. Pozostałe parametry są parametrami wyjściowymi i mają następujące znaczenie:

WR,WI[1:N] - tablice zawierające odpowiednio części rzeczywiste i urojone wartości własnych,

RR,II[1:M1,1:M1] - tablice zawierające odpowiednio części rzeczywiste i urojone prawych wektorów własnych (zapisane wierszami),

VR,VI[1:N,1:M1] - tablice zawierające odpowiednio części rzeczywiste i urojone lewych wektorów własnych (zapisane wierszami),

rA,iA[1:3,1:M1,1:M1] - tablice zawierające odpowiednio części rzeczywiste i urojone macierzy równania flatteru, jej pierwsze i drugie pochodne (zawartość tych tablic, zależna od sposobu rozwiązania flatterowego zagadnienia na wartości własne, jest szczegółowo opisana w pracy [7]).

Oprócz klawiszy 14 i 15, których funkcje zostały opisane w sieci działań algorytmu (tablica 1), w procedurze wykorzystywany jest klawisz o numerze 7. Przy jego włączeniu funkcja F (3.1) jest obliczana na podstawie części urojonych wartości własnych, a w przypadku przeciwnym na podstawie krzywych współczynnika tłumienia konstrukcyjnego g.

Wydruk kontrolny zawiera informacje o każdej iteracji: numer iteracji, wartość współczynnika częstości k i jego odwrotności, aktualną wartość zmiennej standardowej time oraz informacje o funkcjach f_i i funkcji F (3.1). Informacje o funkcjach składowych (części urojone wartości własnych lub współczynniki tłumienia - w zależności od stanu klawisza 7) są drukowane w kolumnach opatrzonych następującymi nagłówkami:

LP - wartość wskaźnika funkcji f_i (funkcje mogą mieć zmienne numery przy przejściu od jednej iteracji do drugiej),

U[KM/H] - prędkość przepływu w kilometrach na godzinę,

G - współczynnik tłumienia konstrukcyjnego,

F[HZ] - częstość drgań,

MI - wartość funkcji f_i,

MI1 - wartość pochodnej f_i',
MI2 - wartość pochodnej f_i''.
Pięć liczb drukowanych po wypełnieniu kolumn w następnym wierszu oznacza:
W - wartość funkcji F,
W1 - wartość pochodnej F',
W2 - wartość pochodnej F",
ARG - wartość wyrażenia podpierwiastkowego we wzorze (3.2) podzielona przez F^2,
1/K - wartość argumentu następnego przybliżenia.
Po znalezieniu każdego zera drukowany jest odpowiadający mu wektor postaci flatteru w kolumnach pod nagłówkami:
RE - części rzeczywiste składowych wektora,
IM - części urojone składowych wektora,
R - moduły składowych wektora,
FI - argumenty składowych wektora.
Litera P w danym wierszu oznacza, że dotyczy on postaci drgań własnych, natomiast litera R wskazuje na stopnie swobody przemieszczeń sztywnych.

7. Uwagi końcowe

Zaproponowana postać (3.1) funkcji F(x) nie jest oczywiście jedyną nadającą się do wyznaczania miejsc zerowych części urojonych wartości własnych. Jednakże podjęcie prac mających na celu zbadanie innych funkcji nie wydaje się uzasadnione choćby z tego względu, że o efektywności przedstawionego algorytmu decyduje głównie metoda iteracyjna. Z drugiej strony nie należy także oczekiwać, że problem automatycznego wyznaczania prędkości krytycznej flatteru zostanie definitywnie rozwiązany za pomocą nowej metody iteracyjnej. Aktualny stan badań i zdobytych doświadczeń w tej dziedzinie zdaje się wskazywać, że właściwa droga prowadzi poprzez odejście od klasycznego równania flatteru w postaci (2.1) w kierunku nowych metod badania stabilności aeroelastycznej.

Literatura

[1] STROUD W.J., DEXTER C.B., STEIN M.: Automated Preliminary Design of Simplified Wing Structures to Satisfy Strenght and Flutter Requirements. NASA TN D-6534, 1971.

[2] O'CONNEL R.F., HASSIG H.J., RADOVCICH N.A.: Study on Flutter Related Computational Procedures for Minimum Weight Structural Sizing of Advanced Aircraft. NASA CR-2607, 1976.

[3] DESMARAIS R.N., BENNET R.M.: An Automated Procedure for Computing Flutter Eigenvalues. J. Aircraft, 11, 2, 75-80, 1974.

[4] BHATIA K.G.: An Automated Method for Determining the Flutter Velocity and the Matched Point. J.Aircraft, 11, 1, 21-27, 1974.

[5] HASSIG H.J.: An Approximate True Damping Solution of the Flutter Equation by Determinant Iteration. J.Aircraft, 8, 11, 885-889, 1971.

[6] RUDISILL C.S., COOPER J.L.: An Automated Procedure for Determining the Flutter Velocity. J.Aircraft, 10, 7, 442-444, 1973.

[7] GRZĘDZIŃSKI J.: Algorytm automatycznego obliczania flatteru przy pomocy zmodyfikowanej metody iteracyjnej Laguerre'a. Prace IPPT, 42/1978.

```
 procedure AUT5(FLAT,b,M1,N,X1,XN,NN,tlr,XO);
value XO,X1,XN;
integer M1,N,NN;
real b,XO,X1,XN,tlr;
procedure FLAT;
begin
  integer NI,NR,NW,NC,NP,IW,IK,i,,k,l,s;
  boolean BOOL;
  real XP,ra,ia,rs,is,zp,zp1,zr,zi,h,g,g1,g2,eps,
       q1,q2,q3,w1,w2;
  array rA,iA[1:3,1:M1,1:M1],WR,WI[1:N],VR,VI[1:N,1:M1],
        RR,II[1:M1,1:M1],R,I[1:N,1:5];
  procedure mult(ar,ai,br,bi,cr,ci);
  value ar,ai,br,bi;
  real ar,ai,br,bi,cr,ci;
  begin
    cr:=ar×br-ai×bi;
    ci:=ar×bi+ai×br;
  end;
  setoutput(3);
  NR:=1;
  NI:=0;
  q3:=w2:=0.0;
  BOOL:=true;
  NW:=if key(14) then -1 else 1;
  format('123');
  print('??KLAWISZE:');
  for i:=0 step 1 until 23 do if key(i) then print(i);
B1: if XO lt X1 or XO gt XN then goto FL3;
  FLAT(b,XO,M1,N,WR,WI,RR,II,VR,VI,rA,iA);
  for l:=1,2 do
  for i:=1 step 1 until N do
  for j:=1 step 1 until N do
  begin
    if l=2 and i ne j then goto EL;
    ra:=ia:=0.0;
    for s:=1 step 1 until M1 do
    begin
```

```
        rs:=is:=0.0;
        for k:=1 step 1 until M1 do
        begin
          zp:=RR[j,k];
          zp1:=II[j,k];
          zr:=rA[l+1,s,k];
          zi:=iA[l+1,s,k];
          rs:=rs+zp×zr-zp1×zi;
          is:=is+zp×zi+zp1×zr;
        end;
        zp:=VR[i,s];
        zp1:=VI[i,s];
        ra:=ra+zp×rs-zp1×is;
        ia:=ia+zp×is+zp1×rs;
      end;
      rA[l,i,j]:=ra;
      iA[l,i,j]:=ia;
EL:  end;
    l:=1;
    q1:=q2:=w1:=0.0;
    for i:=1 step 1 until N do
    begin
      ra:=WR[i];
      ia:=WI[i];
      rs:=is:=0.0;
      for k:=1 step 1 until i-1,i+1 step 1 until N do
      begin
        mult(rA[1,i,k],iA[1,i,k],rA[1,k,i],iA[1,k,i],zr,zi);
        zp:=ra-WR[k];
        zp1:=ia-WI[k];
        h:=1.0/(zp×zp+zp1×zp1);
        rs:=rs+(zr×zp+zi×zp1)×h;
        is:=is+(zi×zp-zr×zp1)×h;
      end;
      R[i,1]:=rA[1,i,i];
      I[i,1]:=iA[1,i,i];
      R[i,2]:=rA[2,i,i]+2.0×rs;
      I[i,2]:=iA[2,i,i]+2.0×is;
```

```
    if key(7) then
    begin
      g:=ia;
      g1:=I[i,1];
      g2:=I[i,2];
    end else
    begin
      g:=ia/ra;
      g1:=(I[i,1]-g×R[i,1])/ra;
      g2:=(I[i,2]-g×R[i,2]-2.0×g1×R[i,1])/ra;
    end;
    l:=l×sign(g);
    w1:=w1+1.0/abs(g);
    rs:=sign(g)/(g×g);
    q1:=q1+rs×g1;
    q2:=q2+rs×(g2-2.0×g1×g1/g);
    R[i,3]:=g;
    I[i,1]:=g1;
    I[i,2]:=g2;
  end;
E2:
  w1:=l/w1;
  q1:=l×w1×w1×q1;
  q2:=2.0×q1×q1/w1+l×w1×w1×q2;
  rs:=q1×q1/(w1×w1)-q2/w1;
  if rs le 0.0 then
  begin
    h:=abs(q1/q2);
    NP:=sign(q1×q3);
  end else
  begin
    h:=1.0/sqrt(rs);
    NP:=sign(w1×w2);
  end;
  w2:=w1;
  q3:=q1;
  NC:=if BOOL then 1 else NC×NP;
  BOOL:=false;
```

```
  if not(key(15) or h lt tlr) then goto E3;
  line(3);
  format('?␣␣␣NR␣ITERACJI␣=112␣␣␣␣␣␣␣␣␣WSPOŁCZYNNIK␣CZESTOSCI␣
         K␣=123.12345␣(1/K␣=123.12345)␣␣␣␣␣␣␣␣␣CZAS␣=1234␣S')
; print(NI,1.0/XO,1,XO,time);
  line(2);
  print('␣␣␣LP␣␣␣U[KM/H]␣␣␣␣␣␣␣␣␣G␣␣␣␣␣␣␣F[HZ]␣␣␣␣␣␣␣␣MI␣␣␣
␣␣␣␣␣␣␣␣MI1␣␣␣␣␣␣␣␣␣␣␣␣MI2');
  line(1);
  for i:=1 step 1 until N do
  begin
    outchar(15);
    format('␣␣␣12␣␣␣123456.1␣␣␣123.123␣456␣␣123.123');
    zp:=1.0/sqrt(abs(WR[i]));
    zp1:=3.6×XO×b×zp×sign(WR[i]);
    print(i,zp1,WI[i]/WR[i],zp/6.2831853072);
    space(3);
    format('+1.123456₁₀+12␣');
    print(R[i,3],I[i,1],I[i,2]);
  end;
  line(2);
  format('+1.123456₁₀+123␣␣');
  print('W=',w1,'W1=',q1,'W2=',q2,'ARG=',rs,'1/K=',XO+h×NW×NC);
E3:
  if h gt tlr then
  begin
    XO:=XO+h×NW×NC;
    NI:=NI+1;
    goto E1;
  end;
  NR:=NR+1;
  BOOL:=true;
  h:=abs(WI[1]);
  IK:=1;
  for i:=2 step 1 until N do
    if abs(WI[i]) lt h then
    begin
      h:=abs(WI[i]);
```

```
      IK:=i;
    end;
  line(2);
  space(53);
  format('POSTAC␣FLATTERU123?');
  print(IK);
  zp:=1.0/sqrt(abs(WR[IK]));
  zp1:=3.6×XO×b×zp×sign(WR[IK]);
  space(29);
  format('U␣=1234.123[KM/H]␣␣␣F␣=123.123[HZ]␣␣␣K␣=12.12345␣␣␣
␣IM␣=-1.12₁₀+123');
  print(zp1,zp/6.2831853072,1.0/XO,h,'??');
  space(40);
  print('RE␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣IM␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣R␣␣␣␣␣␣␣␣␣␣␣␣␣␣FI?
?');
  format('12␣␣␣␣␣1.12345₁₀+123␣␣␣␣1.12345₁₀+123
          ␣␣␣␣1.12345₁₀+123␣␣␣␣1.12345₁₀+123?');
  for i:=1 step 1 until M1 do
  begin
    space(28);
    if i gt N then
    begin
      line(1);
      space(28);
      print('R',i-N);
    end else print('P',i);
    zp:=RR[IK,i];
    zp1:=II[IK,i];
    print(zp,zp1,sqrt(zp×zp+zp1×zp1),if zp=0.0 then
          sign(zp1)×1.570796327 else arctan(zp1/zp)+
          (if zp gt 0.0 then 0.0 else if zp1=0.0 then
          3.1415926536 else sign(zp1)×3.1415926536));
  end;
  if NR gt NN then goto FL3;
  l:=1;
  q1:=q2:=w1:=0.0;
  for i:=1 step 1 until IK-1,IK+1 step 1 until N do
  begin
```

```
    g:=R[i,3];
    g1:=I[i,1];
    g2:=I[i,2];
    l:=l×sign(g);
    w1:=w1+1.0/abs(g);
    rs:=sign(g)/(g×g);
    q1:=q1+rs×g1;
    q2:=q2+rs×(g2-2.0×g1×g1/g);
  end;
  l:=l×sign(I[IK,1]);
  goto E2;
FL3:
end AUT5;?
```