



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Gép- és Terméktervezési Tanszék



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

A vezikulumok mozgását leíró differenciálegyenlet-rendszer vizsgálata

Csáti Zoltán

I. éves MSc gépészmérnök hallgató

Konzulens:

Vadászné dr. Bognár Gabriella

egyetemi docens

Gépészmérnöki és Informatikai Kar

Miskolc

2013

Tartalomjegyzék

1. Bevezetés	3
2. A feladat modelljének származtatása	4
3. A matematikai modell analitikus megoldása	12
3.1. Egzakt megoldások speciális esetekre	12
3.2. Egzakt megoldás a fázissíkon	14
3.3. A megoldás létezése és egyértelműsége	15
4. A matematikai modell kvalitatív vizsgálata fázisdiagramokkal	18
5. A fázisdiagramokat bemutató számítógépes program	25
6. Összefoglalás, további cél	36
7. Függelék	37
7.1. PhasePlaneAnalysis	37
7.2. iconRead	52
8. Hivatkozások	54

1. Bevezetés

A fizikában és a biológiában gyakran előfordulnak olyan jelenségek, amelyek folyadékban merített kis részecskék mozgásának megismerését teszik szükségessé. Einstein 1905-ben megjelent cikkében folyadékban mozgó gömb alakú szilárd részecskére alkotott matematikai modellt [4]. Jeffrey 1922-ben ellipszoid alakú részecskét tekintett [9], elméleti eredményeit később Taylor kísérletileg igazolta [17]. Először Taylor vizsgált olyan esetet, amikor a folyadékban nem szilárd test, hanem egy folyadékcsepp mozog [18]. A mozgásegyenlet felírása során három feltételezéssel élt. A folyadékcsepp olyan kicsi, hogy közel gömb alakú marad, nincs csúszás a csepp felületén és a csepp felületén a csúsztatófeszültségek folytonosak és a két folyadék határán nem ébred normálfeszültség. Később ezeket a modelleket fel tudták használni vezikulumok és vörösvérsejtek vizsgálatára is.

A vezikulum egy zárt foszfolipid membrán (a foszfolipidek kettős lipid réteget képesek alkotni), amely a belsejében lévő folyadékot elválasztja a membránon kívül lévő folyadéktól. A kettős hártya vastagsága körülbelül 5 nm , a határolt térrész jellemző sugara ennek ezerszerese, de léteznek $10\text{--}100\text{ }\mu\text{m}$ nagyságú óriás vezikulumok is [20]. A vezikulum alapvetően különbözik más membrán által határolt testektől, például a polimer kapszuláktól. A vezikulumoknak több egyensúlyi helyzete van és külső áramlás hatására komplexebb nem-egyensúlyi állapotok alakulnak ki. Különbség a kapszulák és a vezikulumok között az, hogy a vezikulumok feszültségi állapota főleg hajlításból származik, míg a polimerizált membránok jellemző igénybevétele nyírás és húzás. Ezen kívül a lipid membránok külső erő hatására úgy végeznek alakváltozást, hogy a felületük állandó maradjon. Továbbá a hőhatás is befolyásolja a vezikulumok dinamikáját, mivel a testek mikrométeres nagyságúak. A lipid hárták biológiai jelentősége miatt egyre növekszik a fizikai érdeklődés irántuk: oxigén transzport, anyagminták és gének célzott testbe juttatása. Fontos szerepük van a vörösvérsejtek vizsgálatában és a membránok biofizikájában. Viszkózus áramlásban elhelyezett vezikulumok dinamikájának tanulmányozása különösen jelentős, mert más – bonyolultabb viselkedésű – sejtek, mint például a vörösvérsejtek viselkedése megérthető ezen keresztül [13]. A modell és a számítás helyessége összehasonlítható mérésekkel (lásd pl. [13], [19]). Ezen kívül a vezikulumok deformációjára vonatkozó modell lépésről lépésre finomítható, így fokozatosan megkaphatjuk a valódi sejtek tulajdonságait. Megfigyelték, hogy viszkózus áramlás hatására a vezikulumoknak három alapvető mozgásformája alakul ki [8]. Az egyik az úgynevezett "tank-treading", amikor a vezikulum ellipszoid alakúvá deformálódik és főtengelye az áramlás irányával Θ szöget zár be, ahol $0 < \Theta < \pi/4$. A második mozgástípus a "tumbling", amikor a membrán merev testszerű forgómozgást végez – kivéve, ha a vezikulum gömb alakú. A fentebbi két mozgás ellipszoid alakú vezikulumokra érvényes és először Keller és Skalak írta le [10]. Létezik egy köztes állapot is, melyet "vacillating-breathing"-nek neveznek és Misbah származtatta [14] cikkében. Ilyen típusú mozgásnál a vezikulum az áramlás iránya mentén rezgő mozgást végez, miközben alakja légző mozgáshoz hasonlóan változik. A továbbiakban ezzel a pontosabb modellel fogunk dolgozni.

Jelen dolgozat célja a Misbah-féle modellből származtatott csatolt közönséges nemlineáris differenciálegyenlet-rendszer vizsgálata és egzakt megoldások keresése a vezikulumok alakjának meghatározására.

2. A feladat modelljének származtatása

Helyezzük a vezikulumot kétdimenziós viszkózus áramlásba, newtoni közeget feltételezve. Az áramlás irányában vegyük fel a koordináta-rendszer x tengelyét, rá merőlegesen az y tengelyt. A zavartalan áramlási sebesség $\mathbf{U}_0 = (\dot{\gamma}y, 0, 0)$, ahol $\dot{\gamma}$ a deformáció sebesség. Az áramlást a vezikulumon kívül és belül a Stokes-egyenletek írják le, mert $Re \ll 1$. A Reynolds-szám akkor lehet kicsi, ha nagyon rövid pályán, vagy kis sebességgel mozognak a folyadékrészek, illetve ha nagy a viszkozitásuk. Jelenleg az utóbbi két eset áll fenn. Ekkor a Navier-Stokes egyenletből a sebességmező idő szerinti teljes deriváltja elhanyagolható a viszkózus erőkhöz képest. Tehát a Stokes-egyenlet és a kontinuitási egyenlet:

$$\mu \Delta \mathbf{u} - \nabla p = \mathbf{0}, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

ahol p az áramló közeg nyomása, $\mathbf{u}$ a sebességmezője, μ pedig a dinamikai viszkozitása. Hasonlóan írhatók fel a vezikulum belsejében lévő folyadékra a Stokes-egyenletek, ahol megkülönböztetésül az $\tilde{\mathbf{u}}$, $\tilde{p}$ és $\tilde{\eta}$ mennyiségeket használjuk.

A Stokes-egyenleteknek Lamb-féle analitikus megoldása a következőképpen állítható elő. A (2.1) divergenciáját véve és felhasználva a (2.2) kontinuitási egyenletet, a nyomásmezőre vonatkozó Laplace-egyenletet kapjuk:

$$\Delta p = 0. \quad (2.3)$$

A vezikulumok ellipszoidhoz közeli alakja miatt a (2.3) differenciálegyenletet gömbi koordináta-rendszerben érdemes vizsgálni. A Descartes-féle (x, y, z) és a gömbi (r, φ, ϑ) koordináták közötti kapcsolat:

$$x = r \cos \varphi \sin \vartheta,$$

$$y = r \sin \varphi \sin \vartheta,$$

$$z = r \cos \vartheta.$$

Ezzel (2.3)-ból kapjuk a következő egyenletet:

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial p}{\partial r} \right) + \frac{1}{r^2 \sin \vartheta} \frac{\partial}{\partial \vartheta} \left(\sin \vartheta \frac{\partial p}{\partial \vartheta} \right) + \frac{1}{r^2 \sin^2 \vartheta} \frac{\partial^2 p}{\partial \varphi^2} = 0.$$

A z -tengelyre való szimmetria esetén az utolsó tag 0, azaz

$$\frac{\partial}{\partial r} \left(r^2 \frac{\partial p}{\partial r} \right) + \frac{1}{\sin \vartheta} \frac{\partial}{\partial \vartheta} \left(\sin \vartheta \frac{\partial p}{\partial \vartheta} \right) = 0. \quad (2.4)$$

Keressük a (2.4) megoldását szétválasztható alakban [5]:

$$p(r, \vartheta) = R(r)\Theta(\vartheta),$$

mellyel két közönséges differenciálegyenletet kapunk:

$$r^2 R'' + 2r R' - n(n+1)R = 0, \quad (2.5)$$

$$\Theta'' + \operatorname{ctg} \vartheta \Theta' + n(n+1)\Theta = 0, \quad (2.6)$$

ahol n állandó. A (2.5) egy Euler-típusú differenciálegyenlet, melynek két lineárisan független megoldása

$$R_1 = r^n, \quad R_2 = r^{-(n+1)}.$$

A (2.6) differenciálegyenletben az $x = \cos \vartheta$ új független változót bevezetve, majd $\Theta(\vartheta) = w(\cos \vartheta)$ -t felhasználva az

$$(1 - x^2) \frac{d^2 w}{dx^2} - 2x \frac{dw}{dx} + n(n+1)w = 0, \quad (n = 0, 1, \dots) \quad (2.7)$$

n -edfokú Legendre-féle differenciálegyenletet kapjuk. A megoldásokat végtelen sor alakjában keresve a $w(x) = P_n(x)$ Legendre-féle polinomokat nyerjük:

$$\begin{aligned} P_0(x) &= 1, \\ P_n(x) &= \frac{1 \cdot 2 \cdot \dots \cdot (2n-1)}{n!} \left[x^n - \frac{n(n-1)}{(2n-1)2} x^{n-2} + \dots + \right. \\ &\quad \left. + (-1)^m \frac{n(n-1) \dots (n-2m+1)}{(2n-1)(2n-3) \dots (2n-2m+1) 2 \cdot 2 \cdot \dots \cdot 2m} x^{n-2m} + \dots \right], \\ &\quad \text{ha } n \neq 0. \end{aligned}$$

Így már felírható a nyomás a vezikulum belsejében

$$\tilde{p}(r, \vartheta) = \sum_{n=0}^{\infty} a_n r^n P_n(\cos \vartheta)$$

és azon kívül

$$p(r, \vartheta) = \sum_{n=0}^{\infty} b_n r^{-n-1} P_{-n-1}(\cos \vartheta).$$

A p illetve $\tilde{p}$ nyomásból kiszámítható az $\mathbf{u}$ és $\tilde{\mathbf{u}}$ sebességmező is, amelynek Lamb-féle megoldása [12]:

$$\begin{aligned} \mathbf{u} &= \sum_{n=0}^{\infty} \nabla \chi_{-n-1} \times \mathbf{r} + \nabla \phi_{-n-1} - \frac{n-2}{2n(2n-1)} r^2 \nabla p_{-n-1} + \frac{n+1}{n(2n-1)} \mathbf{r} p_{-n-1}, \\ \tilde{\mathbf{u}} &= \sum_{n=0}^{\infty} \nabla \tilde{\chi}_n \times \mathbf{r} + \nabla \tilde{\phi}_n + \frac{n+3}{2(n+1)(2n+3)} r^2 \nabla \tilde{p}_n - \frac{n}{(n+1)(2n+3)} \mathbf{r} \tilde{p}_n, \end{aligned}$$

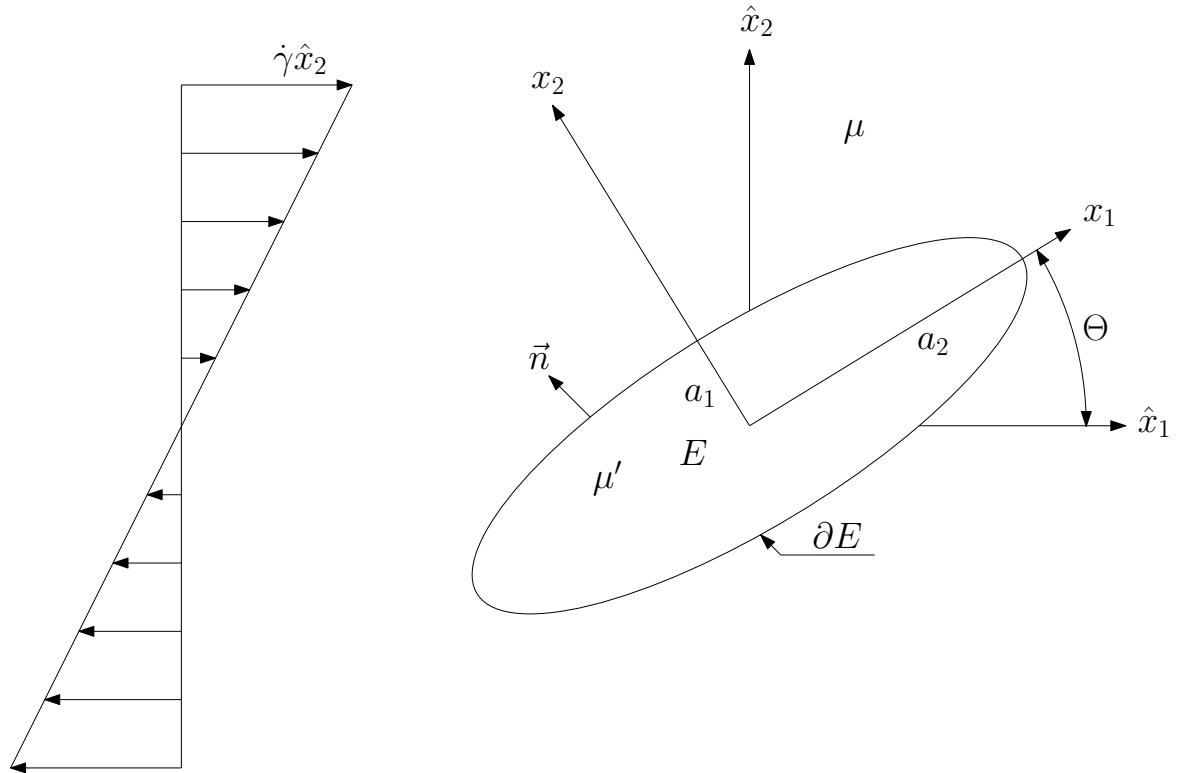
ahol $\tilde{\chi}_n$, $\tilde{\phi}_n$, $\tilde{p}_n$ n -edfokú, χ_{-n-1} , ϕ_{-n-1} , p_{-n-1} pedig $-n-1$ -edfokú gömbi harmonikusok, amelyek már tartalmazzák az a_n és b_n együtthatókat. Ezek a peremfeltételekből határozhatók meg: a sebességmező folytonosságából és a membránra ható erőkből. A megoldásban az első tag állandó nyomású térben örvénylő áramlást fejez ki. A második tag az állandó nyomásra és örvénymentes áramlásra utal. Az utolsó két tag a nyomáseloszlással kapcsolatos, ahol a

nyomásmezőt gömbi harmonikusokkal írtuk fel:

$$p = \sum_{n=0}^{\infty} p_n.$$

Először röviden a bevezetőben említett Keller-Skalak modell származtatását mutatjuk be [10] alapján. Ez a modell a lineáris anyagegyenletű viszkózus áramlásba helyezett ellipszoid alakú membrán és az azon belül lévő folyadék mozgását írja le.

Jelölje 0x_i a kezdeti, x_i a pillanatnyi, $\hat{x}_i$ pedig a referencia koordináta-rendszer koordináta tengelyeit ($i = 1, 2, 3$) és m a membránra (azaz ∂E -re) vonatkozó jellemzőket (lásd 1. ábra).



1. ábra. Viszkózus áramlásba helyezett ellipszis alakú membrán

Síkbeli áramlásról lévén szó ${}^0x_3 = x_3 = \hat{x}_3$. A vezikulum x_3 tengely körüli forgását a $\Theta(t)$ függvény adja meg. Feltéve, hogy az ellipszoid felületének "tank-treading" típusú mozgása van, a membrán sebessége az együtt mozgó koordináta-rendszerben

$$v_1^m = \nu \left(-\frac{a_1}{a_2} \right) x_2, \quad v_2^m = \nu \frac{a_2}{a_1} x_1, \quad v_3^m = 0,$$

ahol ν frekvencia mértékegységgel rendelkezik és általános esetben függ az időtől. A membrán mozgásegyenlete Lagrange-féle leírasi módban:

$$x_1({}^0\mathbf{x}, t) = {}^0x_1 \cos \omega - \frac{a_1}{a_2} {}^0x_2 \sin \omega, \quad x_2({}^0\mathbf{x}, t) = {}^0x_2 \cos \omega + \frac{a_2}{a_1} {}^0x_1 \sin \omega, \quad x_3({}^0\mathbf{x}, t) = {}^0x_3,$$

$$\text{ahol } \omega = \int_0^t \nu(t') dt'.$$

A peremfeltételek a következők: a membránnal kívülről és belülről érintkező folyadékreszc-

kék sebessége megegyezik a membrán sebességével és az ellipszoidtól végtelen távol lévő folyadék áramlási sebessége azonos a zavartalan áramlási sebességgel. A membránon kívüli folyadékról a membránra ható erők és nyomatékok vizsgálatával a Θ szögelfordulásra a következő differenciálegyenlet adódik:

$$\dot{\Theta} = \tilde{A} + \tilde{B} \cos 2\Theta,$$

ahol

$$\tilde{A} = -\left(\frac{1}{2}\dot{\gamma} + \frac{2a_1a_2}{a_1^2 + a_2^2}\nu\right), \quad \tilde{B} = \frac{1}{2}\dot{\gamma}\frac{a_1^2 - a_2^2}{a_1^2 + a_2^2}.$$

A membrán belsejében lévő folyadék esetén kihasználható az energia-megmaradási tétel, azaz a külső folyadék által a membránon végzett munka megegyezik a membránon és a belső folyadékban bekövetkező energia disszipációval. A számítások után a

$$\dot{\Theta} = A + B \cos 2\Theta, \quad (2.8)$$

differenciálegyenletet kapjuk, ahol

$$A = -1/2\dot{\gamma},$$

$$B = 2\dot{\gamma} \left[\frac{1}{2} + \frac{1}{z_1} f_3 \left(\frac{1}{f_2 - \lambda f_1} \right) \right] z_1 \left(\frac{1}{r_2} + r_2 \right)^{-1}.$$

A B -ben szereplő paraméterek jelentése:

$$r_2 = \frac{a_2}{a_1}, \quad r_3 = \frac{a_3}{a_1}, \quad \lambda = \frac{\mu'}{\mu}$$

$$\alpha_1 = r_2^{-1/3} r_3^{-1/3}, \quad \alpha_2 = r_2^{2/3} r_3^{-1/3}, \quad \alpha_3 = r_2^{-1/3} r_3^{2/3},$$

$$z_1 = \frac{1}{2} \left(\frac{1}{r_2} - r_2 \right), \quad z_2 = g'_3 (\alpha_1^2 + \alpha_2^2),$$

$$f_1 = \left(r_2 - \frac{1}{r_2} \right)^2, \quad f_2 = 4z_1^2 \left(1 - \frac{2}{z_2} \right), \quad f_3 = -4 \frac{z_1}{z_2}$$

és g'_1 az ellipszoid alakjától függő integrál. A fenti összefüggésekből látszik, hogy a membrán mozgását a B -beli $\lambda = \mu'/\mu$ viszkozitási hányados és az ellipszoid főtengelyeinek a_2/a_1 , illetve a_3/a_1 aránya befolyásolja.

A (2.8) differenciálegyenlet megoldása $B > -A$ esetén

$$\Theta(t) = \arctg \left[\frac{B + A}{\sqrt{B^2 - A^2}} \frac{D \exp(2\sqrt{B^2 - A^2}t) + 1}{D \exp(2\sqrt{B^2 - A^2}t) - 1} \right], \quad (2.9)$$

alakban adható meg, ahol

$$D = \frac{\sqrt{B^2 - A^2} \operatorname{tg} \Theta_0 + A + B}{\sqrt{B^2 - A^2} \operatorname{tg} \Theta_0 - A - B}, \quad \Theta_0 = \Theta(t = 0).$$

A (2.9)-ből levonható az a következtetés, hogy

$$\lim_{t \rightarrow \infty} \Theta(t) = \Theta^* = \frac{1}{2} \arccos \left(-\frac{A}{B} \right),$$

Mivel $0 \leq -A/B \leq 1$, ezért $0 \leq \Theta^* \leq \pi/4$.

A (2.8) differenciálegyenlet megoldása $B < -A$ paraméterérték esetén:

$$\Theta(t) = \arctg \left[\frac{A+B}{\sqrt{A^2-B^2}} \operatorname{tg} \left(\frac{(t-t_0)\pi}{T} \right) \right],$$

ahol t_0 a $\Theta = 0$ -hoz tartozó időpont és T a 180° -os átfordulás periódusideje:

$$T = (\pi A^2 - B^2)^{-1/2}.$$

A "vacillating-breathing" esetre a Misbah-féle modellt [14] ismertetjük. A membránra ható erők normál komponense ekkor az

$$\mathbf{F}_n = [\kappa [2H(2H^2 - 2K) + 2\Delta_B H] - 2\zeta H] \mathbf{n}$$

módon, míg a tangenciális komponense az

$$\mathbf{F}_t = (\mathbf{I} - \mathbf{n} \circ \mathbf{n}) \cdot \nabla \zeta$$

módon határozhatók meg, ahol κ a membrán hajlító szilárdsága, $H = (1/R_1 + 1/R_2)/2$ a közepes görbület, $K = 1/(R_1 R_2)$ a Gauss-féle görbület, $\zeta(\mathbf{r}^m, t)$ a Lagrange-multiplikátor, ami a membrán helyi összenyomhatatlanságát biztosítja, Δ_B a Laplace-Beltrami-operátor, ami így írható:

$$\Delta_B = \frac{1}{\sqrt{g}} \partial_i (g^{ij} \sqrt{g} \partial_j),$$

ahol $g = \det(g_{ij})$. Továbbá $\mathbf{R}_i$ a görbe vonalú koordináta-rendszer bázisvektorait adja meg indexes jelölésmóddal és g^{ij} a metrikus tenzor kontravariáns komponensei, míg $\mathbf{n}$ az erő normál komponensének egységvektora, $\mathbf{I}$ az egységtenzor. A felületi összenyomhatatlanságot kifejező egyenlet: $(\delta_{ij} - n_i n_j) \partial_i u_j = 0$, ahol n_i a membrán felületi normálisának i -edik komponense. Mostantól a hosszméreteket a vezikulum r_0 egyenértékű sugarára (az a sugár, amely sugarú gömb térfogata megegyezik a vezikulum V térfogatával, azaz $r_0 = (3V/4\pi)^{1/3}$), az időt pedig a deformáció sebesség reciprokára vonatkoztatjuk. Keressük az r értékét (amely a gömbtől való eltérés mértékét mutatja)

$$r = 1 + \epsilon \sum_{n=1}^{\infty} f_n, \quad (2.10)$$

perturbációs sor alakjában, ahol ϵ egy kis paraméter. Az $\mathbf{U}_0$ alakja miatt az egyensúlyi egyenletekben a gömbi harmonikusok sorai csak másodfokkal bezárólag tartalmaznak tagokat. Például

$$f_2 = \sum_{m=-2}^2 F_{2m} \mathcal{Y}_{2m},$$

ahol $\mathcal{Y}_{2m}$ gömbi harmonikusok. A végső differenciálegyenlet származtatása bonyolult, a szerző nem is közölte cikkében. Az F_{22} -re vonatkozó differenciálegyenlet

$$-i\epsilon\partial_t F_{22} = F_{22} - h + 2h\Delta^{-1}(|F_{22}|^2 - F_{22}^2), \quad (2.11)$$

ahol Δ a vezikulum gömbhöz képesti felületeltérése: $\Delta = (A - 4\pi r_0^2)/r_0^2$ és a h paraméterre fennáll: $h = 60\sqrt{2\pi/15}/(32+23\lambda)$. A (2.11)-ben F_{22} -t $Re^{-2i\psi}$ -nek vesszük fel. A (2.11) valós és képzetes részét tekintve megkapjuk a bővített modellre vonatkozó evolúciós egyenleteket, amelyek kis alakváltozásra igazak:

$$\epsilon\frac{\partial R}{\partial t} = h\left[1 - 4\frac{R^2}{\Delta}\right]\sin 2\Theta, \quad (2.12)$$

$$\epsilon\frac{\partial \Theta}{\partial t} = -\frac{1}{2} + \frac{h}{2R}\cos 2\Theta. \quad (2.13)$$

A (2.12) a membrán alakváltozását, a (2.13) pedig a vezikulum áramló közegben bekövetkező szögelfordulását írja le. Érdekesség, hogy folyadékcseppekre ([2], [6]) és kapszulákra [3] vonatkozó modellnél, a perturbációs sorfejtésből ugyanakkora fokszerű tagokat megtartva mint (2.10)-ben, a differenciálegyenletek lineárisak lesznek, míg vezikulumokra nemlineáris rendszert kapunk. Ez a nemlinearitás a lokális felület-összenyomhatatlansági feltételből adódik. További érdekesség, hogy a (2.12) és (2.13) nem függ a C_a kapillaritási számtól ($C_a = \mu\dot{\gamma}r_0^3/\kappa$ és κ a membrán hajlítási merevségét jelöli). Misbah elmélete a $C_a \rightarrow \infty$ -nek felel meg. Ha a sornak több tagját vesszük figyelembe, akkor megjelenik a C_a tényező is. "Tank-treading" esetben a Keller-Skalak-egyenletek megoldása:

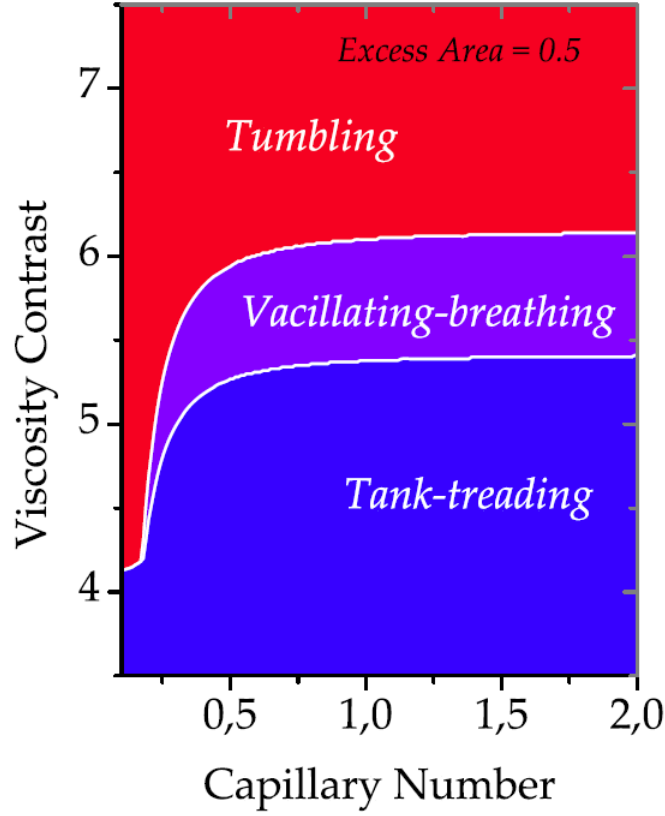
$$R_0 = \frac{\sqrt{\Delta}}{2},$$

$$\Theta_0 = \pm \frac{1}{2} \arccos\left(\frac{\sqrt{\Delta}}{2h}\right).$$

Ebből következik, hogy $\sqrt{\Delta}/(2h) < 1$, vagy $\lambda < \lambda_c$, ahol

$$\lambda_c = -\frac{32}{23} + \frac{120}{23}\sqrt{\frac{2\pi}{15\Delta}}$$

a kritikus viszkozitási arány, amely elválasztja egymástól a "tank-treading" és "tumbling" régiót. Ha $\lambda < \lambda_c$ – vagy ami ezzel egyenértékű: $h < h_c = \sqrt{\Delta}/2$ –, akkor a "tank-treading" és "vacillating-breathing" mozgástípus egyszerre van jelen. Ekkor lineáris stabilitásvizsgálattal kimutatták, hogy a $\Theta_0 = 0$, $R_0 = h$ kritikus pont környezetében zárt fázisgörbék vannak, tehát (2.12), (2.13)-ra periodikus megoldás létezik. Ha $\lambda > \lambda_c$, akkor nyeregponthoz hasonló típusú bifurkáció után egy másik mozgás, a "tumbling" alakul ki. A három típus kialakulását a 2. ábrán látható $\lambda - C_a$ diagram szemlélteti.



2. ábra. A három mozgás kialakulásának feltételei a λ és C_a tekintetében [20]

A (2.12), (2.13) helyett vizsgáljuk az általánosabb

$$\frac{dR}{dt} = (\alpha - \beta R^2) \sin 2\Theta, \quad (2.14)$$

$$\frac{d\Theta}{dt} = -\frac{q}{2} + \frac{\alpha}{2R} \cos 2\Theta \quad (2.15)$$

differenciálegyenlet-rendszert, ahol $\alpha, \beta, q \in \mathbb{R}^+$. A továbbiakban a deriváltak a t változó szerinti deriváltakat jelölik. Bevezetve az új $x(t)$ és $y(t)$ változókat az

$$x(t) = R \cos 2\Theta,$$

$$y(t) = R \sin 2\Theta$$

alakban, a (2.14), (2.15) differenciálegyenletekből az

$$\begin{aligned} x' &= y(q - \beta x) \\ y' &= \alpha - qx - \beta y^2 \end{aligned} \quad (2.16)$$

differenciálegyenlet-rendszert nyerjük.

Ahhoz, hogy a "vacillating-breathing" mozgást tudjuk vizsgálni, fenn kell állnia az

$$R \leq \frac{\sqrt{\Delta}}{2}$$

egyenlőtlenségnek, azaz – felhasználva a bevezetett α és β paramétereket – a következő feltétel

adódik:

$$x^2 + y^2 \leq \frac{\alpha}{\beta}. \quad (2.17)$$

A (2.16) csatolt közönséges nemlineáris differenciálegyenlet-rendszerre egzakt megoldást keresünk, melynek számos előnye van a numerikus megoldással szemben. Az egzakt megoldás az adott fizikai probléma mélyebb megértését teszi lehetővé, amely így bonyolultabb modellek kiindulásaként szolgálhat. Továbbá az egzakt megoldás ismeretében a paraméterek hatása egyszerűbben vizsgálható. Végül pedig tesztproblémaként szolgálhat a közelítő analitikus vagy numerikus sémák teszteléséhez.

3. A matematikai modell analitikus megoldása

3.1. Egzakt megoldások speciális esetekre

Keressük a (2.16) megoldását az

$$y = ax + b \quad (3.1)$$

egyenes mentén. A (3.1)-et behelyettesítve (2.16)-be, az alábbi két egyenletet nyerjük:

$$x' = qb + (aq - \beta b)x - a\beta x^2 \quad (3.2)$$

$$x' = \frac{1}{a} (\alpha - \beta b^2 - (q + 2ab\beta)x - \beta a^2 x^2). \quad (3.3)$$

A (3.2) és a (3.3) jobb oldala csak akkor lehet egyenlő, ha a megfelelő együtthatók megegyeznek. Ezért

$$\begin{aligned} qb &= \frac{\alpha - \beta b^2}{a}, \\ aq - \beta b &= -\frac{q + 2ab\beta}{a}, \end{aligned}$$

melyekből q -ra

$$\begin{aligned} q &= \frac{\alpha - \beta b^2}{ab} \\ q &= -\frac{ab\beta}{1 + a^2}. \end{aligned}$$

A q paramétert kiküszöbölve kapjuk az

$$\frac{\alpha}{\beta} = \frac{b^2}{1 + a^2}$$

összefüggést. Ennek segítségével a (3.2) differenciálegyenlet az alábbi módon írható:

$$x' = -a\beta x^2 + \left(\frac{\alpha}{b} - 2b\beta\right)x + \frac{\alpha - \beta b^2}{a}. \quad (3.4)$$

A (3.4) differenciálegyenlet egy általános alakú Riccati-típusú differenciálegyenlet, amelyre egzakt megoldás általában nem adható. Viszont a paraméterek megfelelő megválasztásával elemi módszerekkel megoldható differenciálegyenletre tudjuk visszavezetni. Az alábbiakban négy esetet különböztetünk meg:

1. Ha $\alpha = \beta b^2$

Ekkor a (3.4) egyenletből a következő Bernoulli-típusú differenciálegyenletet nyerjük:

$$x' + \beta bx = -a\beta x^2. \quad (3.5)$$

Ennek megoldása:

$$x(t) = \left(Ce^{\beta bt} - \frac{a}{b}\right)^{-1}. \quad C \in \mathbb{R}$$

2. Ha $\alpha = 2\beta b^2$

Most a (3.4) egyenlet az

$$x' = -a\beta x^2 + \frac{\beta b^2}{a}$$

változóiban szétválasztható típusú differenciálegyenletté egyszerűsödik. Ennek megoldása:

$$x(t) = \frac{b}{a} \operatorname{th}(\beta b t + M). \quad M \in \mathbb{R}$$

3. Ha $\beta = 0$

A 2. esethez hasonlóan egy szétválasztható differenciálegyenletre jutunk:

$$x' = \frac{\alpha}{b}x + \frac{\alpha}{a}.$$

Ennek megoldása:

$$x(t) = N e^{\frac{\alpha}{b}t} - \frac{b}{a}. \quad N \in \mathbb{R}$$

4. Az a , b , α , β tetszőleges, a (2.17)-nek eleget tevő konstansok

Bár az általános $x'(t) = P(t)x^2 + Q(t)x + R(t)$ Riccati-egyenletnek általában nincs egzakt megoldása, ha csak nem találunk hozzá egy partikuláris megoldást. A (3.4) egyenletben szereplő együtthatók állandók és az $R(t)$ tag is konstans. Ebben az esetben transzformációval a Riccati-egyenlet egy állandó együtthatós másodrendű lineáris differenciálegyenletté transzformálható (lásd [15]). Tekintsük az

$$x' = Px^2 + Qx + R \tag{3.6}$$

speciális esetet, amely a (3.4), ha $P = -a\beta$, $Q = \frac{\alpha}{b} - 2b\beta$ és $R = \frac{\alpha - \beta b^2}{a}$. Alkalmazzuk az

$$x(t) = -\frac{w'(t)}{Pw(t)}$$

helyettesítést. Így az

$$x' = -\frac{1}{P} \left(-\frac{w'^2}{w^2} + \frac{w''}{w} \right),$$

kifejezést behelyettesítve (3.6)-ba egy lineáris másodrendű differenciálegyenletet nyerünk:

$$w'' - Qw' + PRw = 0. \tag{3.7}$$

Ezáltal a (3.7) egyenlet a következő alakba írható:

$$w'' - \left(\frac{\alpha}{b} - 2b\beta \right) w' + \beta(\beta b^2 - \alpha)w = 0,$$

amely általános megoldása

$$w(t) = C_1 e^{\left(\frac{\alpha}{b} - \beta b\right)t} + C_2 e^{-\beta b t}. \quad C_1, C_2 \in \mathbb{R}$$

Legyen $w(0) = 1$, ekkor $C_2 = 1 - C_1$. Visszatérve az eredeti $x(t)$ függvényre, az általános

megoldás így írható:

$$x(t) = \frac{1}{a\beta} \frac{C_1 \left(\frac{\alpha}{b} - \beta b\right) e^{\left(\frac{\alpha}{b} - \beta b\right)t} + (C_1 - 1)\beta b e^{-\beta b t}}{C_1 e^{\left(\frac{\alpha}{b} - \beta b\right)t} + (1 - C_1)e^{-\beta b t}}.$$

3.2. Egzakt megoldás a fázissíkon

Redukáljuk a (2.16) differenciálegyenlet-rendszert egy elsőrendű differenciálegyenletre. Legyen az $x(t)$ függvény invertálható. Ekkor a láncszabály és az inverz függvény deriválási szabálya szerint

$$\frac{dy(t(x))}{dx} = \frac{dy}{dt} \frac{dt}{dx} = \frac{dy/dt}{dx/dt} = \frac{y'}{x'} = \frac{\alpha - qx - \beta y^2}{y(q - \beta x)} \quad (y \neq 0, x \neq \frac{q}{\beta}), \quad (3.8)$$

mely az

$$\underbrace{(\alpha - qx - \beta y^2)}_{U(x,y)} dx + \underbrace{y(\beta x - q)}_{V(x,y)} dy = 0. \quad (3.9)$$

alakba írható fel. Megmutatjuk, hogy a (3.9) egzakt típusú differenciálegyenletté alakítható át. Ahhoz, hogy a (3.9) bal oldala teljes differenciál legyen, azaz legyen olyan $G(x, y)$ függvény, melyre $\partial G/\partial x = U$ és $\partial G/\partial y = V$ teljesül, a

$$\frac{\partial U}{\partial y} = \frac{\partial V}{\partial x}$$

egyenlőségnek kell fennállnia. Mivel

$$\frac{\partial U}{\partial y} = -2\beta y \neq \frac{\partial V}{\partial x} = \beta y,$$

ezért (3.9) nem egzakt. Keressünk egy $k(x, y)$ integrál szorzót, amellyel (3.9) már egzakttá tehető, azaz

$$\frac{\partial(Uk)}{\partial y} = \frac{\partial(Vk)}{\partial x},$$

melyből

$$U \frac{\partial k}{\partial y} - V \frac{\partial k}{\partial x} + k \left(\frac{\partial U}{\partial y} - \frac{\partial V}{\partial x} \right) = 0. \quad (3.10)$$

A (3.10) elsőrendű lineáris parciális differenciálegyenlet egyszerűsödik, ha $k(x, y)$ speciális alakú. Keressünk egy csak x -től függő integráló szorzót. Ekkor (3.10)-ből kapjuk:

$$-V \frac{dk}{dx} + k \left(\frac{\partial U}{\partial y} - \frac{\partial V}{\partial x} \right) = 0,$$

melyből

$$k(x) = e^{\int \frac{1}{V} \left(\frac{\partial U}{\partial y} - \frac{\partial V}{\partial x} \right) dx}.$$

Ez akkor teljesül k -ra, ha a jobb oldalon az integrandusz csak x függvénye. Jelenleg

$$\frac{1}{V} \left(\frac{\partial U}{\partial y} - \frac{\partial V}{\partial x} \right) = \frac{3\beta}{q - \beta x},$$

így

$$k(x) = e^{\int \frac{3\beta}{q-\beta x} dx} = \frac{1}{|q - \beta x|^3}. \quad (3.11)$$

A (3.9) és (3.11) felhasználásával keresett $G(x, y)$ függvényre

$$\frac{\partial G}{\partial x} = \frac{\alpha - qx - \beta y^2}{|q - \beta x|^3}, \quad (3.12)$$

$$\frac{\partial G}{\partial y} = \frac{y(\beta x - q)}{|q - \beta x|^3}. \quad (3.13)$$

Legyen $q > \beta x$. Ekkor a (3.13) egyenletből kapjuk, hogy

$$G(x, y) = -\frac{y^2}{2(q - \beta x)^2} + H(x). \quad (3.14)$$

Deriválva $G(x, y)$ -t y szerint és felhasználva (3.12)-t:

$$H'(x) = \frac{\alpha - qx}{(q - \beta x)^3}.$$

Innen

$$H(x) = \left(\frac{\alpha}{2\beta} - \frac{q^2}{2\beta^2} \right) (q - \beta x)^{-2} + \frac{q}{\beta^2} (q - \beta x)^{-1}.$$

Ezt behelyettesítve (3.14)-be, majd (3.15) alapján a (3.9) általános $G(x, y) = C$ megoldása implicit alakban:

$$\frac{q}{\beta^2} \frac{1}{q - \beta x} + \frac{\alpha\beta - q^2 - \beta^2 y^2}{2\beta^2} \frac{1}{(q - \beta x)^2} = C. \quad (3.15)$$

Könnyen ellenőrizhető, hogy $q < \beta x$ esetén szintén a (3.15) megoldást kapjuk.

3.3. A megoldás létezése és egyértelműsége

Ebben az alfejezetben a (2.16) differenciálegyenlet megoldhatóságát vizsgáljuk adott kezdeti feltételek mellett. Az

$$\begin{aligned} x' &= y(q - \beta x), & x(0) &= x_0 \\ y' &= \alpha - qx - \beta y^2, & y(0) &= y_0 \end{aligned} \quad (3.16)$$

kezdetiérték-feladatot mátrix alakban írjuk fel:

$$\underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\mathbf{z}'} = \underbrace{\begin{bmatrix} 0 & q \\ -q & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\mathbf{z}} + \underbrace{\begin{bmatrix} -\beta xy \\ -\beta y^2 \end{bmatrix}}_{\mathbf{n}} + \underbrace{\begin{bmatrix} 0 \\ \alpha \end{bmatrix}}_{\mathbf{m}}.$$

Bontsuk fel az $\mathbf{A}$ mátrixot egy diagonális $\mathbf{D}$ és egy másik $\mathbf{B}$ mátrix összegére. Majd legyen $\mathbf{F} = \mathbf{Bz} + \mathbf{n}$, amellyel a

$$\mathbf{z}' = \mathbf{Dz} + \mathbf{F} + \mathbf{m} \quad (3.17)$$

mátrix differenciálegyenletet kapjuk, ahol

$$\mathbf{B} = \begin{bmatrix} -1 & q \\ -q & -1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} -x + qy - \beta xy \\ -qx - y - \beta y^2 \end{bmatrix}, \quad \mathbf{z}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}.$$

A (3.16) kezdeti feltétele a

$$\mathbf{z}(0) = \mathbf{z}_0 \quad (3.18)$$

módon írható.

Először megmutatjuk, hogy a (3.17),(3.18) kezdetiérték-feladatnak egy megoldása van [11].

Legyen $I \in [0, T]$, $T < \infty$ és jelölje $C(I)$ az I -n folytonos függvények halmazát, a normát a $\|\mathbf{z}\| = \max_{t \in I} |z_i|$ módon értelmezve. Olyan $\mathbf{z}$ megoldást veszünk figyelembe, amelyre a $\int_0^t \cdot ds$

operátor korlátos: $\exists T > 0$, hogy $\|\int_0^t \mathbf{z} ds\| \leq T\|\mathbf{z}\|$. Tartozzon a $\mathbf{z}$ korlátos függvény a $\mathcal{B}$ Banach-térhez: $\mathcal{B} = \{\mathbf{z} \in \mathbb{R}^2, \|\mathbf{z}\| < d\}$.

1. Lemma. A (3.17)-ben lévő $\mathbf{D}$ lineáris operátor korlátos, $\mathbf{F}$ nemlineáris operátor pedig Lipschitz-folytonos.

Bizonyítás. $\mathbf{D}$ korlátos ha $\exists K > 0$, hogy $\|\mathbf{D}\mathbf{z}\| \leq K\|\mathbf{z}\|$. Mivel a $\mathbf{D}$ egy egységmátrix, ezért $\mathbf{D}$ valóban korlátos $K = 1$ legkisebb felső korláttal. Az $\mathbf{F}$ eleget tesz a Lipschitz-feltételnek az $L > 0$ Lipschitz-konstanssal, ha fennáll a $\|\mathbf{F}(\mathbf{z}_1) - \mathbf{F}(\mathbf{z}_2)\| \leq L\|\mathbf{z}_1 - \mathbf{z}_2\|$, azaz

$$\left\| \begin{array}{l} -x_1 + qy_1 - \beta x_1 y_1 - (-x_2 + qy_2 - \beta x_2 y_2) \\ -qx_1 - y_1 - \beta y_1^2 - (-qx_2 - y_2 - \beta y_2^2) \end{array} \right\| \leq \left\| \begin{array}{l} x_1 - x_2 \\ y_1 - y_2 \end{array} \right\|.$$

Ezzel egyenértékű ha belátjuk, hogy az alábbi négy eset teljesül:

$$|F_1(\mathbf{z}_1) - F_1(\mathbf{z}_2)| \leq L_1|x_1 - x_2|, \quad (|x_1 - x_2| > |y_1 - y_2|), \quad (3.19)$$

$$|F_1(\mathbf{z}_1) - F_1(\mathbf{z}_2)| \leq L_2|x_1 - x_2|, \quad (|x_1 - x_2| < |y_1 - y_2|), \quad (3.20)$$

$$|F_2(\mathbf{z}_1) - F_2(\mathbf{z}_2)| \leq L_3|x_1 - x_2|, \quad (|x_1 - x_2| > |y_1 - y_2|), \quad (3.21)$$

$$|F_2(\mathbf{z}_1) - F_2(\mathbf{z}_2)| \leq L_4|x_1 - x_2|, \quad (|x_1 - x_2| < |y_1 - y_2|). \quad (3.22)$$

A (3.19) fennáll $L_1 = 1 + q + 2d\beta$ állandóval, ugyanis

$$\begin{aligned} |F_1(\mathbf{z}_1) - F_1(\mathbf{z}_2)| &= |-x_1 + qy_1 - \beta x_1 y_1 - (-x_2 + qy_2 - \beta x_2 y_2)| \leq \\ &\leq |x_1 - x_2| + q|y_1 - y_2| + \beta[|x_2||y_2 - y_1| + |y_1||x_2 - x_1|] \leq \\ &\leq |x_1 - x_2| + q|x_1 - x_2| + \beta 2d|x_1 - x_2| = \\ &= (1 + q + 2d\beta)|x_1 - x_2| = \\ &= L_1|x_1 - x_2|. \end{aligned}$$

A (3.20) teljesen hasonló, de ott $|x_1 - x_2| < |y_1 - y_2|$ használható fel, így (3.20) is igaz $L_2 = L_1$ konstanssal. Ebből a megfontolásból a (3.21) és (3.22) közül csak a (3.21)-et nézzük, amelyre

szintén igaz az egyenlőtlenség $L_3 = L_1$ -re.

$$\begin{aligned}
|F_2(\mathbf{z}_1) - F_2(\mathbf{z}_2)| &= -qx_1 - y_1 - \beta y_1^2 - (-qx_2 - y_2 - \beta y_2^2) \leq \\
&\leq q|x_1 - x_2| + |y_1 - y_2| + \beta|y_1 - y_2||y_1 + y_2| \leq \\
&\leq q|x_1 - x_2| + |x_1 - x_2| + \beta 2d|x_1 - x_2| = \\
&= (1 + q + 2d\beta)|x_1 - x_2| = \\
&= L_1|x_1 - x_2|.
\end{aligned}$$

Azaz teljesül a globális Lipschitz-feltétel. □

1. Tétel. A (3.17), (3.18) kezdetiérték-feladatnak pontosan egy $\mathbf{z}$ megoldása létezik, ha $0 < \gamma < 1$, ahol $\gamma = (K + L)T$.

Bizonyítás. Írjuk át a (3.17) differenciálegyenletet integrálegyenletté a (3.18) kezdeti feltételt alkalmazva:

$$\mathbf{z} = \mathbf{z}_0 + \int_0^t [\mathbf{D}\mathbf{z}(s) + \mathbf{F}(\mathbf{z}(s)) + \mathbf{m}(s)] ds.$$

Definiáljuk az $\mathcal{F} : C(I) \rightarrow (C(I))^2$ operátort az alábbi módon:

$$\mathcal{F}(\mathbf{z}) = \mathbf{z}_0 + \int_0^t [\mathbf{D}\mathbf{z}(s) + \mathbf{F}(\mathbf{z}(s)) + \mathbf{m}(s)] ds.$$

$$\begin{aligned}
\|\mathcal{F}(\mathbf{z}) - \mathcal{F}(\mathbf{z}^*)\| &= \left\| \int_0^t [\mathbf{D}(\mathbf{z} - \mathbf{z}^*) + \mathbf{F}(\mathbf{z}) - \mathbf{F}(\mathbf{z}^*)] ds \right\| \leq \\
&\leq \int_0^t \|\mathbf{D}(\mathbf{z} - \mathbf{z}^*) + \mathbf{F}(\mathbf{z}) - \mathbf{F}(\mathbf{z}^*)\| ds \leq \\
&\leq \int_0^t [\|\mathbf{D}(\mathbf{z} - \mathbf{z}^*)\| + \|\mathbf{F}(\mathbf{z}) - \mathbf{F}(\mathbf{z}^*)\|] ds \leq \\
&\leq \int_0^t [K\|\mathbf{z} - \mathbf{z}^*\| + L\|\mathbf{z} - \mathbf{z}^*\|] ds \leq \\
&\leq (K + L) \int_0^t \|\mathbf{z} - \mathbf{z}^*\| ds \leq \\
&\leq (K + L)T\|\mathbf{z} - \mathbf{z}^*\| = \\
&= \gamma\|\mathbf{z} - \mathbf{z}^*\|.
\end{aligned}$$

Az egyenlőtlenség két végét összehasonlítva: $(1 - \gamma)\|\mathbf{z} - \mathbf{z}^*\| \leq 0 \Rightarrow \|\mathbf{z} - \mathbf{z}^*\| = 0 \Rightarrow \mathbf{z} = \mathbf{z}^*$. Tehát $\mathcal{F}$ kontrakció I -n, ezért egyetlen fix pontja van, így pontosan egy $\mathbf{z}$ megoldás létezik. □

4. A matematikai modell kvalitatív vizsgálata fázisdiagramokkal

Az (2.16) differenciálegyenlet-rendszer vizsgálatához fázistér analízist végzünk. Vegyünk egy általános autonóm differenciálegyenlet-rendszert vektoros alakban:

$$\mathbf{y}' = \mathbf{f}(\mathbf{z}), \quad (4.1)$$

ahol $\mathbf{f} : D \rightarrow \mathbb{R}^n$, $D \subseteq \mathbb{R}^n$. A vektoregyenlet megoldása az a $\Phi(t)$ függvény, amely differenciálható, és kielégíti (4.1)-t. Ennek geometriai képe egy egyparaméteres görbesereg (trajektóriák). A görbék növekvő idő szerint irányítottak, nem metszhetik egymást és különböző kezdeti feltételekhez más-más ponton áthaladó görbe tartozik [1]. A (4.1) egyensúlyi/kritikus pontja az a pont D -ben, ahol az $\mathbf{f}(\mathbf{z})$ vektormező zérus értéket vesz fel: $\mathbf{f}(\mathbf{z}) = \mathbf{0}$. Két egyenletből álló autonóm differenciálegyenlet-rendszer skaláris alakja:

$$\begin{aligned} x'(t) &= f(x(t), y(t)) \\ y'(t) &= g(x(t), y(t)). \end{aligned} \quad (4.2)$$

Ebben az esetben jól alkalmazható a fázistér analízis, amikor az $x(t)$ és $y(t)$ megoldást egymás függvényében ábrázoljuk. Ekkor kapjuk meg a fázisportrét. Legyen $P(x_0, y_0)$ a (4.2) kritikus pontja. Az $\mathbf{f}(\mathbf{z})$ vektor-vektor függvény $P = \mathbf{z}_0$ pont körüli Taylor sora lineáris közelítésben:

$$\mathbf{f}(\mathbf{z}) \approx \mathbf{f}(\mathbf{z}_0) + \nabla \mathbf{f}(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0} (\mathbf{z} - \mathbf{z}_0),$$

ahol $\nabla \mathbf{f} \equiv \mathbf{J}$ a Jacobi-mátrix, amely a $\mathbf{J} = [\partial f_i / \partial z_j]_{i,j=1}^n$ módon számítható, a gradiens értelmezése alapján. Mivel P kritikus pont, ezért a fentebbi közelítés:

$$\mathbf{f}(\mathbf{z}) \approx \mathbf{J}(\mathbf{z}_0)(\mathbf{z} - \mathbf{z}_0). \quad (4.3)$$

A (4.1) nemlineáris rendszer csak strukturális stabilitás esetén jellemezhető a $\mathbf{z}' = \mathbf{A}\mathbf{z}$ lineáris differenciálegyenlet-rendszerrel. Egy rendszer akkor strukturálisan stabil, ha megfelelően kicsiny perturbáció hatására a perturbált differenciálegyenlet ekvivalens az eredeti differenciálegyenlettel [7]. Ekkor a kritikus pontok az alábbiak közül az egyik csoportba esnek: fókusz, csomópont, nyeregpont. Ezek megállapítása a Jacobi-mátrix sajátértékeinek alapján történik. Ha a sajátértékek

- valósak és ellentétes előjelűek: nyeregpont (instabil)
- valósak, különbözőek és azonos előjelűek: csomópont
 - mindkettő negatív: nyelő csomópont (aszimptotikusan stabil)
 - mindkettő pozitív: forrás csomópont (instabil)
- komplexek, valós részük nem nulla: fókusz
 - a valós rész negatív: nyelő fókusz (aszimptotikusan stabil)
 - a valós rész pozitív: forrás fókusz (instabil)

A forrás azt jelenti, hogy a megoldások a kritikus ponttól távolodnak, a nyelő pedig, hogy közelednek. A nyelő minden esetben aszimptotikusan stabil kritikus pontot eredményez, míg a forrás instabil, más néven labilis pontot. Ha a fenti feltételek nem teljesülnek (ezt nevezik elfajuló esetnek), hanem a sajátértékek

- egyenlőek és
 - két lineárisan független sajátvektor van: nem elfajuló csomópont
 - * a sajátértékek negatívak: nyelő (aszimptotikusan stabil)
 - * a sajátértékek pozitívak: forrás (instabil)
 - egy lineárisan független sajátvektor van: elfajuló csomópont
 - * a sajátértékek negatívak: nyelő (aszimptotikusan stabil)
 - * a sajátértékek pozitívak: forrás (instabil)
- tisztán képzetes komplex számok: centrum (stabil)

Ilyenkor a linearizált differenciálegyenlet-rendszerből nem következtethetünk a nemlineáris differenciálegyenlet-rendszer viselkedésére. Létezik egy szemléletes módszer a stabilitás vizsgálatára. Ez az úgynevezett determináns-nyom diagram, ami könnyen származtatható. Határozzuk meg az

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

mátrix sajátértékeit. A karakterisztikus polinom $p(r) = \det(\mathbf{A} - r\mathbf{I})$ módon írható ($\mathbf{I}$ az egység-mátrix), melyből

$$p(r) = r^2 - (a + d)r + (ad - bc) = r^2 - r \operatorname{tr} \mathbf{A} + \det \mathbf{A},$$

ahol $\operatorname{tr} \mathbf{A}$ az $\mathbf{A}$ mátrix nyoma, $\det \mathbf{A}$ pedig a determinánsa. A következőkben elvégezzük a fázistér vizsgálatot, melynek menete a következő.

1. Meghatározzuk az egyensúlyi/kritikus pontokat
2. Kiértékeljük a Jacobi-mátrixot az egyensúlyi pontokban
3. A linearizált differenciálegyenlet-rendszert sajátérték feladatra vezetjük vissza és a sajátértékek és sajátvektorok segítségével vázoljuk a fázisportrét.

1. Kritikus pontok

Megoldandó a következő egyenletrendszer:

$$\begin{aligned} y(q - \beta x) &= 0 \\ \alpha - qx - \beta y^2 &= 0. \end{aligned} \tag{4.4}$$

Innen három kritikus pont adódik, P_1 , P_2 és P_3 , melyek koordinátái:

$$P_1 \left(\frac{\alpha}{q}, 0 \right), \quad P_2 \left(\frac{q}{\beta}, \frac{1}{\beta} \sqrt{\alpha\beta - q^2} \right), \quad P_3 \left(\frac{q}{\beta}, -\frac{1}{\beta} \sqrt{\alpha\beta - q^2} \right). \quad (4.5)$$

2. Kiszámítjuk a Jacobi-mátrix helyettesítési értékeit a kritikus pontokban

A Jacobi-mátrix:

$$J = \begin{bmatrix} -\beta y & q - \beta x \\ -q & -2\beta y \end{bmatrix}.$$

Ide behelyettesítve a P_1 , P_2 és P_3 pontokat:

$$J|_{P_1} = \begin{bmatrix} 0 & q - \alpha\beta/q \\ -q & 0 \end{bmatrix},$$

$$J|_{P_2} = \begin{bmatrix} -\sqrt{\alpha\beta - q^2} & 0 \\ -q & -2\sqrt{\alpha\beta - q^2} \end{bmatrix},$$

$$J|_{P_3} = \begin{bmatrix} \sqrt{\alpha\beta - q^2} & 0 \\ -q & 2\sqrt{\alpha\beta - q^2} \end{bmatrix}.$$

A mátrixok elemeiből az vehető észre, hogy az $\alpha\beta = q^2$, az $\alpha\beta < q^2$ és az $\alpha\beta > q^2$ esetek a lényegesek. A (4.3) szerinti linearizált egyenletrendszer, bevezetve a $\mathbf{z} := \mathbf{z} - \mathbf{z}_0$ jelölést: $\mathbf{z}' = \mathbf{J}\mathbf{z}$. Keressük a megoldást a

$$\mathbf{z} = \mathbf{v}e^{rt} \quad (4.6)$$

alakban, ahol $\mathbf{v} \in \mathbb{R}^2$ konstans vektor és $r \in \mathbb{R}$. A (4.6) összefüggést alkalmazva, a

$$(\mathbf{J} - r\mathbf{I})\mathbf{v} = \mathbf{0} \quad (4.7)$$

homogén lineáris egyenletrendszert kapjuk, melynek a triviálistól eltérő megoldása akkor lehet, ha az együttható mátrixa szinguláris: $\det(\mathbf{J} - r\mathbf{I}) = 0$. A 3. pontban a 2×2 -es mátrix sajátértékeit, a 4. pontban a sajátvektorait határozzuk meg.

3. Sajátértékek meghatározása

(a) Ha $\alpha\beta > q^2$

Vezessük be a $\alpha\beta - q^2 = \epsilon$ jelölést.

i. a P_1 pontban

$$\begin{vmatrix} -r & -\epsilon/q \\ -q & -r \end{vmatrix} = 0 \rightarrow r_1 = \sqrt{\epsilon}, \quad r_2 = -\sqrt{\epsilon}$$

Mivel r_1 és r_2 valós és eltérő előjelű, ezért P_1 egy nyeregpont.

ii. a P_2 pontban

$$\begin{vmatrix} -\sqrt{\epsilon} - r & 0 \\ -q & -2\sqrt{\epsilon} - r \end{vmatrix} = 0 \rightarrow r_1 = -\sqrt{\epsilon}, \quad r_2 = -2\sqrt{\epsilon}$$

Mivel r_1 és r_2 valós és negatív, ezért P_2 egy aszimptotikusan stabil csomópont.

iii. a P_3 pontban

$$\begin{vmatrix} \sqrt{\epsilon} - r & 0 \\ -q & 2\sqrt{\epsilon} - r \end{vmatrix} = 0 \rightarrow r_1 = \sqrt{\epsilon}, r_2 = 2\sqrt{\epsilon}$$

Mivel r_1 és r_2 valós és pozitív, ezért P_3 egy labilis csomópont.

(b) Ha $\alpha\beta = q^2$

A P_2 és P_3 pontok ordinátái (lásd (4.5)) ilyen paraméter értékek esetén nullák, és a három pont abszcisszája is megegyezik, ugyanis

$$\frac{\alpha}{q} - \frac{q}{\beta} = \frac{\alpha\beta - q^2}{q\beta} = 0,$$

tehát ekkor a három pont egybeesik. Jelen esetben $\alpha\beta - q^2 = \epsilon = 0$, ezért $r_1 = r_2 = 0$. Ez a $\det \mathbf{J} - \text{tr } \mathbf{J}$ diagram origója, így perturbáció hatására bármely fentebb írt stabilitási pont előállhat. Ilyenkor az ábrázolás sem egyszerű, a 4. fejezetben bemutatott programmal oldottuk meg.

(c) Ha $\alpha\beta < q^2$

Vezessük be az előzőekhez hasonlóan $\epsilon - 1$ -szeresét: $\bar{\epsilon} = q^2 - \alpha\beta$. A (4.4) egyetlen valós megoldása a P_1 -et adja.

i. a P_1 pontban

$$r^2 + \bar{\epsilon} = 0 \rightarrow r_1 = \sqrt{\bar{\epsilon}}i, r_2 = -\sqrt{\bar{\epsilon}}i$$

Mivel $\Re(r_1) = \Re(r_2) = 0$, ezért a P_1 pont egy stabil centrum.

4. Sajátvektorok meghatározása

Mivel nem a linearizált differenciálegyenlet-rendszer általános megoldását keressük, hanem a ábrázolás a célunk, ezért a sajátvektorokat csak az $\alpha\beta > q^2$ és a $\alpha\beta < q^2$ esetben határozzuk meg, a $\alpha\beta = q^2$ esetben nem segítene a fázisportré vázolásában. A sajátértékeket beírva a (4.7)-be, 2-2 sajátvektort kapunk.

(a) Ha $\alpha\beta > q^2$

i. a P_1 pontban

$$-qv_1^{(1)} - \sqrt{\epsilon}v_2^{(1)} = 0 \xrightarrow{v_1^{(1)}:=1} v^{(1)} = \begin{bmatrix} 1 \\ -q/\sqrt{\epsilon} \end{bmatrix},$$

a másik sajátvektor:

$$-qv_1^{(2)} + \sqrt{\epsilon}v_2^{(2)} = 0 \xrightarrow{v_1^{(2)}:=1} v^{(2)} = \begin{bmatrix} 1 \\ q/\sqrt{\epsilon} \end{bmatrix}$$

ii. a P_2 pontban

$$-qv_1^{(1)} - \sqrt{\epsilon}v_2^{(1)} = 0 \xrightarrow{v_1^{(1)}:=1} v^{(1)} = \begin{bmatrix} 1 \\ -q/\sqrt{\epsilon} \end{bmatrix},$$

a másik sajátvektor:

$$\sqrt{\epsilon}v_1^{(2)} + 0v_2^{(2)} = 0 \xrightarrow{v_1^{(2)}:=0, v_2^{(2)} \text{ tetsz.}} v^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

iii. a P_3 pontban

$$-qv_1^{(1)} - \sqrt{\epsilon}v_2^{(1)} = 0 \xrightarrow{v_1^{(1)}:=1} v^{(1)} = \begin{bmatrix} 1 \\ -\frac{q}{\sqrt{\epsilon}} \end{bmatrix},$$

a másik sajátvektor:

$$-\sqrt{\epsilon}v_1^{(2)} + 0v_2^{(2)} = 0 \xrightarrow{v_1^{(2)}:=0, v_2^{(2)} \text{ tetsz.}} v^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

(b) Ha $\alpha\beta < q^2$

i. a P_1 pontban

$$\begin{bmatrix} -\sqrt{\bar{\epsilon}}i & \bar{\epsilon}/q \\ -q & -\sqrt{\bar{\epsilon}}i \end{bmatrix} \begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

melyből

$$-\sqrt{\bar{\epsilon}}v_1^{(1)} + \frac{\bar{\epsilon}}{q}v_2^{(1)} = 0 \xrightarrow{v_1^{(1)}:=1} v^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + i \begin{bmatrix} 0 \\ q/\sqrt{\bar{\epsilon}} \end{bmatrix}$$

A másik sajátvektor:

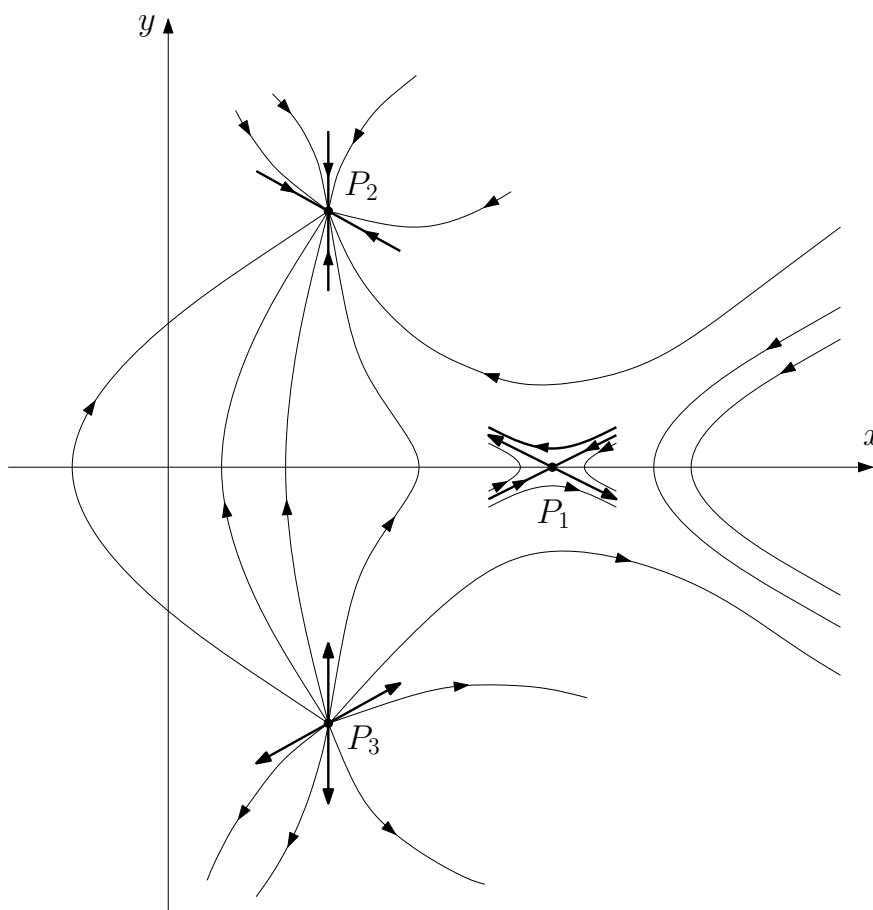
$$\begin{bmatrix} \sqrt{\bar{\epsilon}}i & \bar{\epsilon}/q \\ -q & \sqrt{\bar{\epsilon}}i \end{bmatrix} \begin{bmatrix} v_1^{(2)} \\ v_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

melyből

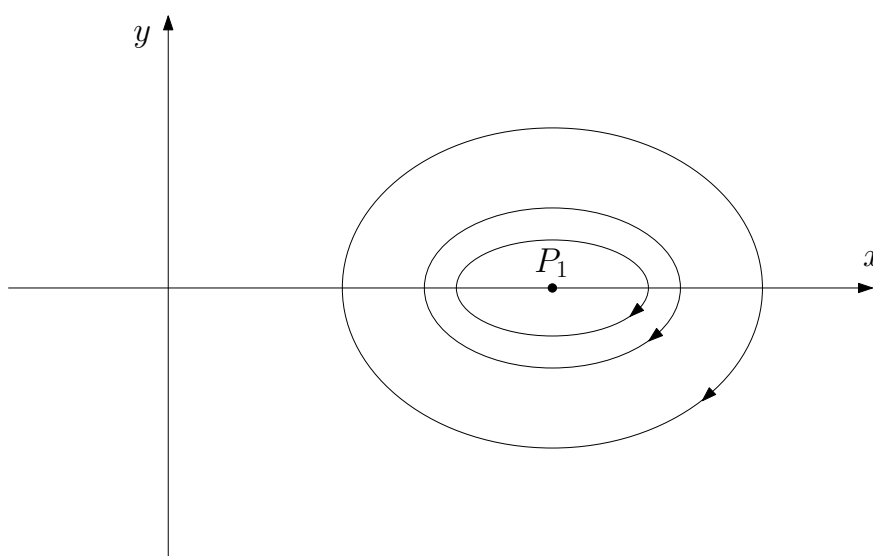
$$v^{(2)} = i \begin{bmatrix} 1 \\ -q/\sqrt{\bar{\epsilon}} \end{bmatrix}.$$

5. Fázisportrék vázolása

A fentiek segítségével már vázolható a fázisportré az $\alpha\beta > q^2$ (3. ábra) és a $\alpha\beta < q^2$ (4. ábra) esetén. A 3. és a 4. ábrákból észrevehető, hogy a (2.16) megoldásai a fázissíkon az x -tengelyre szimmetrikusak. A következő fejezetben számítógépes grafikus ábrázolásból láthatjuk, hogy az $\alpha\beta = q^2$ esetén is fennáll a szimmetria.



3. ábra. A fázisportré $\alpha\beta > q^2$ esetén



4. ábra. A fázisportré $\alpha\beta < q^2$ esetén

Vizsgáljuk most azt, hogy létezik-e határciklus. Erre egy elégséges feltételt ad a Dulac-kritérium [7].

2. Tétel. *Ha létezik egy $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ folytonosan differenciálható függvény továbbá $\nabla \cdot (hf)$ ($f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$) folytonos és nem zérus egy egyszeresen összefüggő D tartományon, akkor nem létezik határciklus D -n.*

Az (2.16) differenciálegyenlet-rendszerre igaz a következő állítás:

3. Tétel. Ha $\beta \neq 0$, akkor nem létezik határciklus $D \subseteq \mathbb{R}^2$ -n.

Bizonyítás. Legyen $g(x, y) = 1$ és vegyük a (4.2), mint vektortér divergenciáját.

$$\mathbf{f} = \begin{bmatrix} y(q - \beta x) \\ \alpha - qx - \beta y^2 \end{bmatrix},$$

$$\operatorname{div} \mathbf{f} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = -3\beta y,$$

amely csak akkor 0, csak ha $y = 0$, de az x -tengely nem egy 2D-s tartomány. $\square$

A következőkben vizsgáljuk meg a (2.17) második feltételét, amely szerint a megoldásoknak egy origó középpontú $\sqrt{\alpha/\beta}$ sugarú körlapra kell esniük. Ha a kritikus pontok is ezen a körön belül vannak, melynek sugara 1-nél nagyobb ($\alpha > \beta$), akkor az alábbiaknak kell fennállni. Ahhoz, hogy P_1 a körön belül legyen, $\alpha/q < \sqrt{\alpha/\beta}$ kell teljesüljön, melyből $\alpha\beta < q^2$. Tehát csak a paraméterek szempontjából vizsgált harmadik eset tartozik ide. A P_2 és P_3 pontok – függetlenül q -tól – mindig a kör kerületén vannak, mert

$$\sqrt{\frac{q^2}{\beta^2} + \frac{\alpha\beta - q^2}{\beta^2}} = \sqrt{\frac{\alpha}{\beta}}$$

a kör sugara.

5. A fázisdiagramokat bemutató számítógépes program

A (4.2) differenciálegyenlet-rendszer fázisképének elemzésére a MATLAB® numerikus programcsomag felhasználásával egy PhasePlaneAnalysis nevű programot írtunk.¹ A MATLAB-ben nincs beépített eszköz ilyen célú vizsgálatra. A készített program egy grafikus felhasználói felület (GUI), amelynek futtatásához az alap MATLAB-en kívül a Symbolic Math Toolbox (SMT) szükséges, azért, hogy a kritikus pontokat ne a felhasználónak kelljen megadnia. Külön figyelmet fordítottunk arra, hogy a program kompatibilis legyen korábbi MATLAB verziókkal, ezért nem használtunk beágyazott függvényeket, és a *save* utasítást is úgy adtuk meg, hogy az előző verziók is be tudják tölteni a szükséges adatokat. A program az alábbi funkciókra képes:

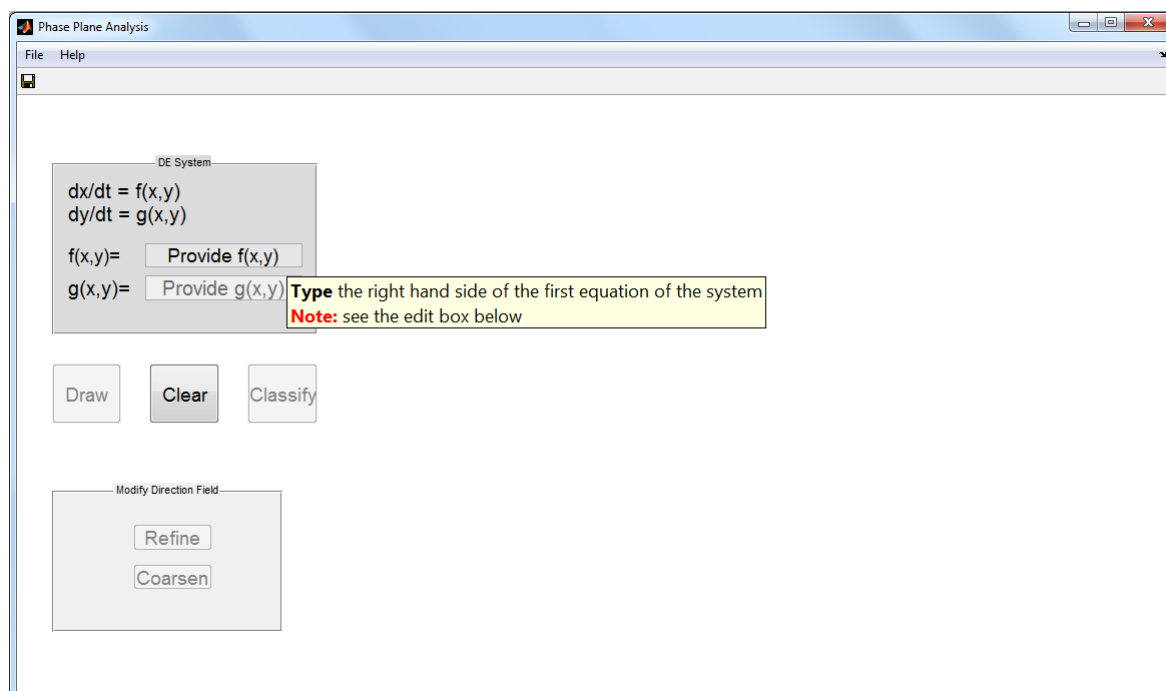
- A (4.2) kritikus pontjainak megkeresése
- A (4.2) differenciálegyenlet-rendszer fázisképének ábrázolása a kritikus pontok környezetében
- A vonalelemek számának növelése vagy csökkentése
- Kritikus pontok osztályozása a Jacobi-mátrix sajátértékei alapján
- A (4.2) differenciálegyenlet-rendszerhez tartozó kezdeti feltételeket a felhasználó adhatja meg egérgattintással. Az így kapott kezdetiérték-feladatot a MATLAB beépített megoldói oldják meg
- Beállítható a megoldó típusa, az integrálási tartomány, és egy biztonsági opció is. Ez utóbbi biztosítja, hogy a megoldási folyamat fejeződjön be, ha a számítási időtartam meghaladja az öt másodpercet. A hosszú számítási idő leggyakrabban akkor lép fel, ha túl nagy az integrálási tartomány, vagy ha a differenciálegyenlet merev. Ilyen esetben tehát nem kell kilépni a programból, elég a megoldási feltételeken változtatni.
- Az integrálgörbék kirajzolásakor változtatható a vonal stílusa, szélessége, színe és a kiszámított pontokat jelölő alakzatok (*marker*)
- A készített grafikát elmenthetjük

Most a grafikus felület megvalósításáról írunk röviden. Két módszer létezik GUI-k létrehozására a MATLAB-ben. Az egyik a GUIDE (Graphical User Interface Development Environment), ami egy interaktív elrendezést segítő alkalmazás. A másik módszer az, hogy a felületet magunk programozzuk. Mi az utóbbit választottuk, mert nagyobb szabadságot tesz lehetővé. A MATLAB-ben létrehozhatunk lokális, globális vagy *persistent*-nek nevezett változót. A lokális változót csak az a függvény látja, amelyikhez tartozik. Ha *persistent* típusúnak deklaráljuk a változót egy függvényen belül, akkor – a lokális változóhoz hasonlóan – csak az a függvény látja, amelyikhez tartozik, de az újabb függvényhívásokkor emlékszik az előző értékére, mert a memóriában marad. Így alkalmas időzítők létrehozására. Ha egy változót globálisnak deklarálunk, akkor minden függvény munkalapról, sőt a parancssorból is látható. Ez veszélyes,

¹Ezentúl a saját MATLAB függvényeket `typewriter` stílussal, a MATLAB utasításokat és a beépített függvényeket *dőlt* betűvel szedjük.

ezért ennek használatát elkerültük. Az egyes függvény munkalapok között többféleképpen lehet adatokat megosztani. Továbbíthatja egy másik függvény argumentumként, alkalmazhatunk beágyazott függvényeket, *persistent* vagy pedig globális változókat. A GUI-k – a számos *callback* függvény miatt – sok függvényt tartalmaznak. A *callback* azt adja meg, hogy mi történjen akkor ha valamilyen esemény következik be az adott objektumon. Speciálisan a GUI-k esetén használható a *setappdata/getappdata* páros, valamint a *guidata* parancs is adattárolásra-és kinyerésre. Az összes grafikus objektum *handle*-jét egy *S*-sel jelölt struktúrában raktározzuk. A *handle* segítségével lehet egy függvényt indirekt módon – nem a függvény nevével, hanem azonosítójával – hívni. A következő részben a program általános bemutatása következik.

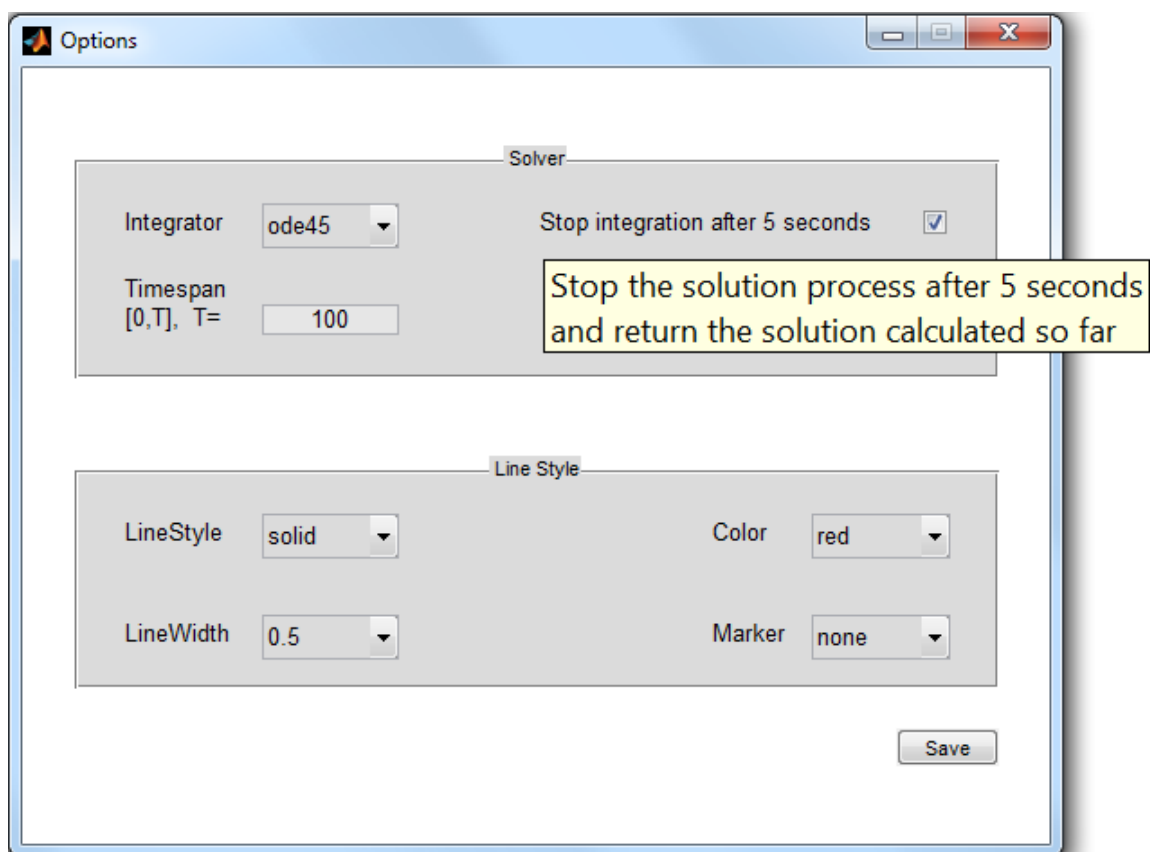
Ha a *PhasePlaneAnalysis* függvényt paraméterek nélkül hívjuk meg, akkor a SMT megpróbálja megkeresni a kritikus pontokat. Ha a SMT nem találja meg a kritikus pontokat, akkor lehetőség van arra, hogy manuálisan adjuk meg. Ilyenkor két bemeneti paramétert adunk meg. Mindkét paraméter egy n dimenziós vektor (n a kritikus pontok száma) – a pontok x és y koordinátái. Ha megtörtént a függvény meghívása, akkor a 5. ábrán látható grafikus felületet láthatjuk.



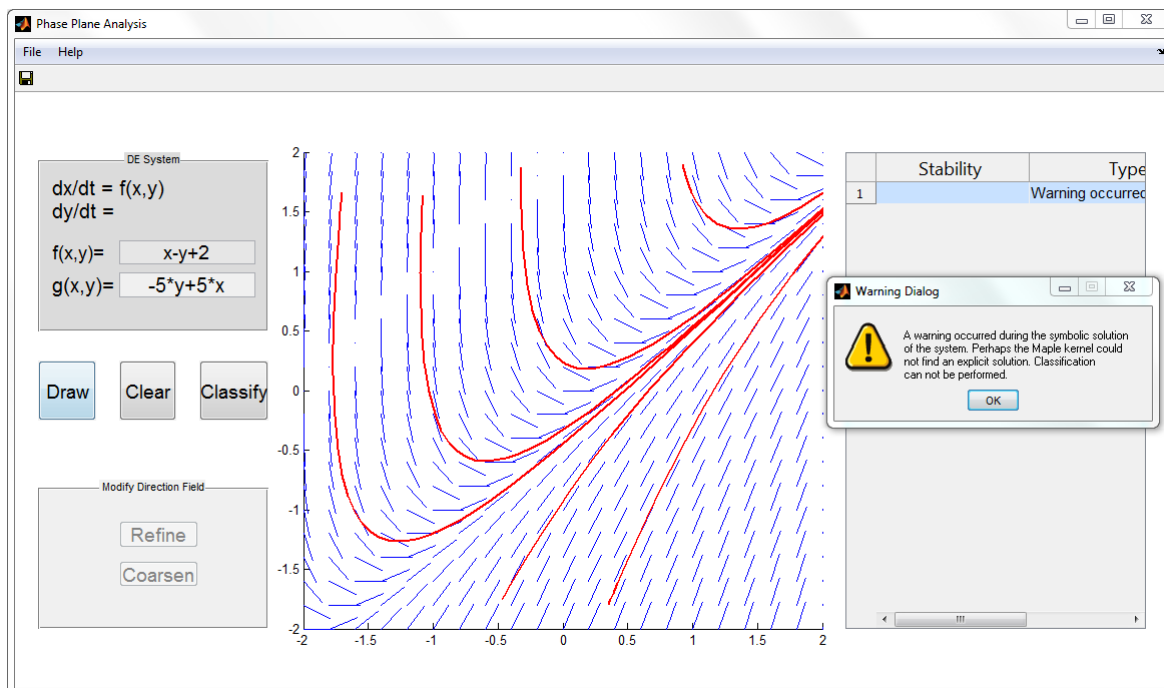
5. ábra. A fő ablak közvetlenül a megnyitás után

A használatban segít a *Help* → *Usage* menüpont és a kurzornál megjelenő szövegbuborékok, ha azt a kérdéses hely felé mozdítjuk. Az eszköztáron lévő floppy ikonnal az ábrát menthetjük el *CurrentFigure.bmp* névvel a MATLAB pillanatnyilag használt könyvtárába. Ha más névvel, más formátumban vagy másik mappába akarjuk menteni, akkor azt a *File* → *Save as...* menünél tehetjük meg. A *File* → *Options...* helyen a (4.2) rendszerhez tartozó kezdetiérték-feladatot megoldó függvény beállításai adhatók meg, amelyek akkor lépnek érvénybe, ha megnyomtuk a *Save* nyomógombot. A beállításokat két részre választottuk. A *Solver* panelen a numerikus megoldóra, míg a *Line Style* panelen a kapott közelítő megoldás ábrázolására vonatkozó jellemzők adhatók meg.

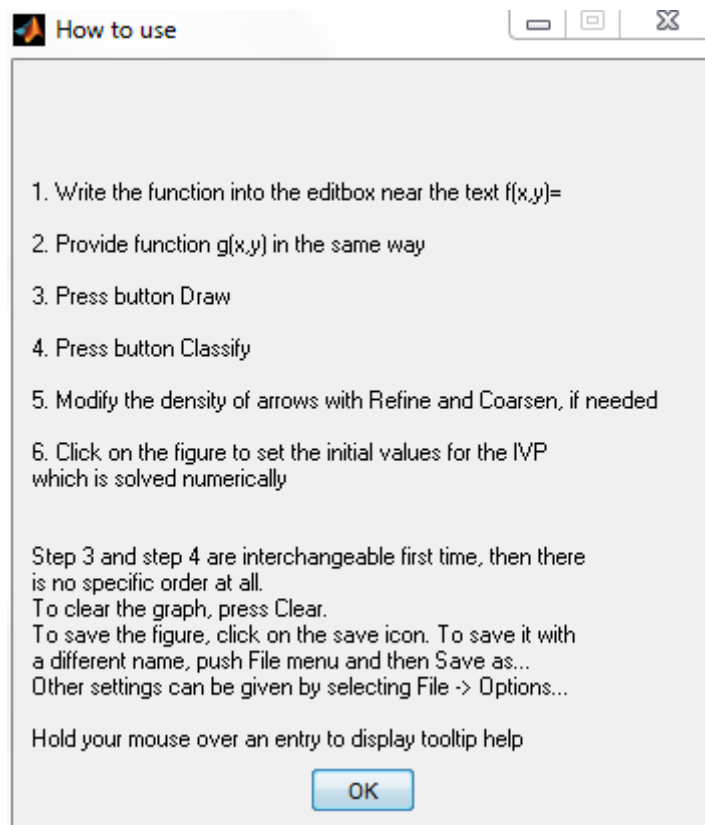
Az alábbi ábrák a programból kerültek kivágásra.



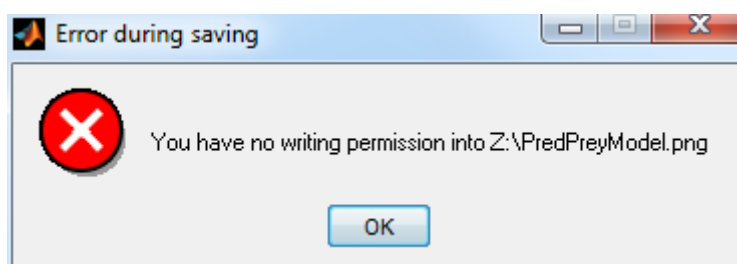
6. ábra. Az *Options* ablak



7. ábra. Figyelmeztetés, ha nincs egyértelmű megoldás



8. ábra. Használati utasítás



9. ábra. Hibaablak mentés esetén

Bár a program jól dokumentált, a jellegzetességeit az 1. táblázatban mutatjuk be. A bal oldali oszlopban a Függelékben található `PhasePlaneAnalysis.m` fájl soraira utalunk, mellette a leírás látható.

Sorok száma	Megjegyzések
1	Függvény definíció.
2-24	A program általános leírása.
28-60	Általános hibaellenőrzés. A MATLAB hibával tér vissza, ha nincs telepítve a Symbolic Math Toolbox. Ezután a bemenő paramétereket vizsgálja. Vagy 0 input van, vagy pedig 2. Ez utóbbi esetben két azonos méretű vektornak kell lenniük.

63-84	A GUI alapját adó ábra és tengelyek létrehozása. A szokásos menüket letiltjuk, kezeljük hogy mi történjen, ha bezárjuk az ablakot, beállítjuk a méretet és a háttérszínt. A tengelyeket egyelőre láthatatlanná tesszük, mert jelenleg még nem áll rendelkezésre adat ahhoz, hogy az ábrába kattintva a program elvégezze a kívánt műveletet, így a <i>callback</i> hibát jelezze. Ezután az <i>S</i> struktúrába tesszük az egyensúlyi pontok koordinátáit, feltéve hogy a felhasználó paraméterekkel hívta meg a <i>PhasePlaneAnalysis</i> függvényt.
86-155	Panelek, nyomógombok, statikus és szerkeszthető szövegdobozok létrehozása. Itt még csak a pozíciójukat, a méretüket, megnevezésüket és a gyermek-szülői viszonyt állítjuk be. Később fogjuk megadni azt, hogy mi történjen akkor, ha valamilyen esemény következik be az adott objektumon. Erre szolgál a <i>callback</i> függvény megadása. A szövegbuborékokban a szöveg, illetve a táblázat fejléce hagyományos módon nem formázható, ezért HTML formázást használtunk. Minden esetben a szövegek méretét és az objektumok pozícióját relatív módon (a szülő objektumhoz képest) adtuk meg, így bármilyen felbontáson a felhasználói felület arányos.
157-167	A menüsorban létrehozunk két menüt – a <i>File</i> és <i>Help</i> menüt – és ezek almenüjeit.
169-187	Azt a táblázatot hozzuk létre – egyelőre nem látható – amelyben a kritikus pontok stabilitását, típusát és koordinátáit jelenítjük meg.
189-194	Az eszköztáron létrehozuk az ábra mentésére szolgáló ikont. Ahhoz, hogy a mentés ikon képe megjelenjen, a MATLAB <i>Product Help</i> -jében példaként bemutatott <i>iconRead</i> függvényt használjuk.
196-209	A grafikus objektumokhoz <i>callback</i> függvényeket állítunk be. Ezeknek a függvényeket később definiáljuk. Egy <i>callback</i> -et meg lehet adni sztringként, <i>function handle</i> -ként, cella tömbként vagy ezek kombinációjaként. Nézzünk rá példákat a programból. A <i>set(S.mainWindow, ... 'DeleteFcn','close('all'))</i> ; esetén azt állítottuk be, hogy az <i>S.mainWindow handle</i> -l rendelkező objektum (a fő ablak) ha megszűnik (bezárjuk), akkor az összes ablak záródjon be. Ilyen egyszerű esetben elegendő sztringként megadni a <i>callback</i> függvényt. A <i>set(S.FileMenu_saveAs, 'Callback',@saveAs)</i> ; beállításnál <i>function handle</i> -t alkalmaztunk. A bonyolultabb eseményekre példa: <i>set(S.edit_g, 'Callback',{@obtain_g,S})</i> ;. A cella első eleme az <i>obtain_g</i> függvény <i>function handle</i> -ként megadva, a második elem a <i>callback</i> függvény harmadik bemenő paramétere. Az első két paramétert akkor is meg kell adni, ha nem használjuk, ezek gyakori nevükön <i>hObject</i> és <i>eventdata</i> . Az <i>hObject</i> a <i>callback</i> függvényt meghívó objektum neve, az <i>eventdata</i> pedig az esemény, amit opcionálisan előidéz.
211-220	Egy struktúrát hozunk létre hét mezőnévvel, melyek az alapértelmezett kezdetiérték-feladat megoldási és ábrázolási beállításokat tartalmazzák. A struktúrát <i>options</i> néven .mat fájlként mentjük el úgy, hogy 6-os verziójú MATLAB is meg tudja nyitni.

226-317	Egy másik ablakot hozunk létre <i>Options</i> névvel és létrehozuk rajta a 6. ábrán látható grafikus kezelőket. Korábban már bemutattuk, hogy melyik mire szolgál, így azt nem részletezzük.
319-340	Az <i>Options</i> -ben megadott beállításokat a <i>Save</i> nyomógombra kattintva lehet elmenteni.
342-730	A fő GUI-hoz tartozó <i>callback</i> függvények megadása. A függvényeket külön részletezzük a lentebbi sorokban.
343-346	Engedélyezi az <i>edit_g</i> szövegdobozba való karakterbevitelt.
348-374	Összegyűjtjük a (4.2) jobb oldalát az <i>eq_f</i> és <i>eq_g</i> változókba. Ha létezik az <i>S</i> struktúrának <i>solution</i> nevű mezője (a <i>PhasePlaneAnalysis</i> függvénynek két bemeneti paramétere van), akkor azt használjuk fel, ha nincs, akkor a <i>solveSystem</i> függvénnyel megkeressük a kritikus pontokat. A koordinátákat és az esetleges figyelmeztetést egy struktúrába rakjuk, amelyet az <i>S.edit_g UserData</i> mezőjébe teszünk. Minden grafikus objektum rendelkezik a <i>UserData</i> mezővel. Itt tetszőleges változó típust tárolhatunk, de egyszerre csak egyet. Ezek után engedélyezzük a <i>Draw</i> és a <i>Classify</i> gombokat.
376-488	A <i>Classify</i> gomb lenyomására a <i>classifyPoints</i> függvény végzi az egyensúlyi pontok osztályozását. A <i>data = get(S.edit_g,'UserData')</i> ; beolvassa a <i>data</i> változóba a fentebb említett <i>UserData</i> -ban tárolt struktúrát. Ha ennek a <i>warning</i> mezője üres sztring (azaz nem lépett fel figyelmeztető üzenet a szimbolikus megoldás során), akkor végrehajtja a legkülső <i>if</i> utasításmagját. Itt először meghatározzuk a Jacobi-mátrixot a <i>computeJac</i> segítségével, majd létrehozuk a <i>J</i> -vel jelölt 3 dimenziós tömböt, amely <i>nroSolution</i> darab (kritikus pontok száma) 2×2 -es mátrixként fogható fel. Ebben tároljuk a Jacobi-mátrix egyensúlyi pontokban felvett helyettesítési értékeit. Lefoglalunk még egy <i>pointType</i> nevű <i>nroSolution</i> $\times 4$ méretű cellát is, ezt fogjuk feltölteni a for-ciklusban, majd pedig kiíratni a 7. ábra jobb oldalán látható világoskék táblázatba. Az táblázat első oszlopába kerül a három lehetséges stabilitási eset, a másodikba a kritikus pont típusa, a harmadik és negyedik oszlopába pedig az adott pont <i>x</i> és <i>y</i> koordinátája. Elvégezzük pontonként a kritikus pontok besorolását, végül pedig láthatóvá tesszük a táblázatot. Az első <i>elseif</i> akkor lép életbe, ha a Jacobi-mátrix nullmátrix. Ha hiba lépett fel a szimbolikus megoldás során, de a hibaüzenet azt jelzi, hogy a mátrix szinguláris, így nincs egyértelmű megoldás, akkor a különböző elfajuló esetek besorolását végzi el a függvény. Ha egyéb hiba keletkezik, akkor a külső <i>else</i> magja fut le és egy figyelmeztető üzenetet kapunk, ahogy ezt a 7. ábra mutatja.

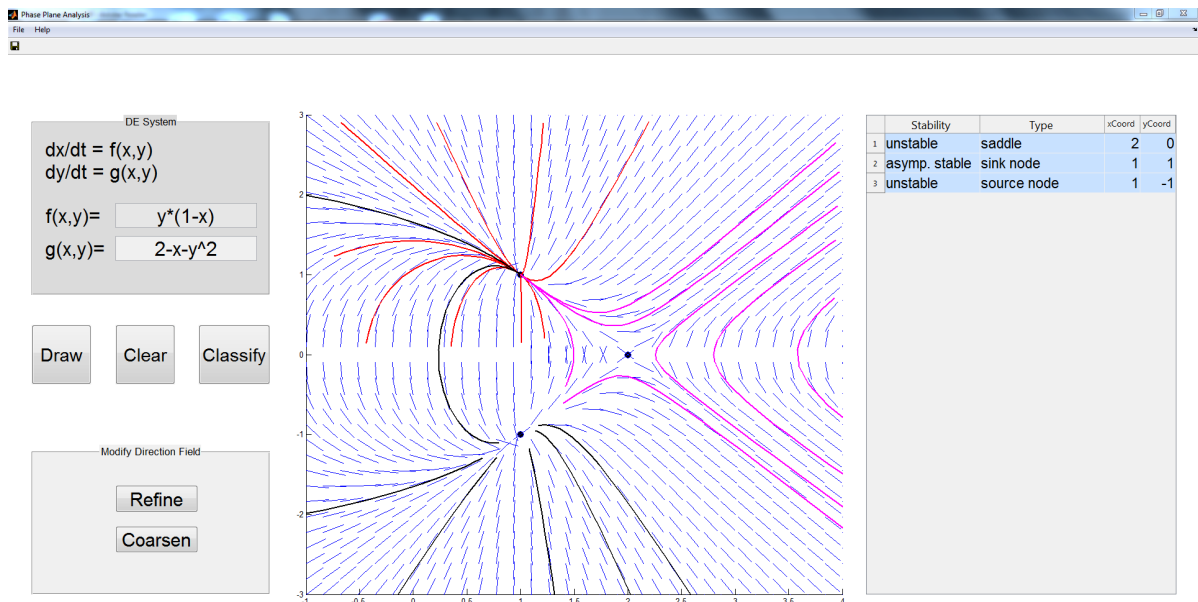
490-553	A <code>Draw</code> gomb lenyomásával és felengedésével aktiválódik a <code>sketchDirfield</code> függvény. Ha van korábbi ábrázolás, akkor azt törli, hogy egyszerre ne legyen több sűrűségű iránymező. Beolvassuk az egyensúlyi pontokat. Ha a <code>warning</code> mező nem üres (tehát nem kaptunk egyértelmű megoldást az algebrai egyenletrendszerre), akkor az alapértelmezett $(x, y) \in [-2, 2] \times [-2, 2]$ tartományban ábrázoljuk az iránymezőt. Ellenkező esetben a tartományt úgy választjuk meg, hogy a legkisebb és legnagyobb x illetve y koordinátájú pontok köré egy 2 szélességű sávot teszünk. Utána elhelyezzük a kritikus pontokat jelölő tömör karikákat, majd pedig a (3.8) szerint előállítjuk dy/dx -et, majd a <code>dirfield</code> -del elvégezzük az iránymező ábrázolását. Ha nem fordult elő hiba az egyenletrendszer megoldásakor, akkor elérhetővé tesszük a <code>Refine</code> és <code>Coarsen</code> nyomógombokat és az eddig láthatatlan tengelyeket.
555-560	A <code>restoreBlank</code> végzi az ábra törlését.
562-585	Az <code>increaseDensity</code> a vonalelemek számának növelésével az iránymező pontosabb ábrázolását biztosítja. Az ötlet amire épül: $nInterval = dataFromDrawDirectionField.nInterval + 2$; , tehát az eddig jelenlévő sűrítést növeli.
587-606	A <code>decreaseDensity</code> hasonló felépítésű az <code>increaseDensity</code> -vel, azonban ez a vonalelemek számát csökkenti.
608-663	A <code>initializeIVP</code> a (4.2)-hez társított kezdetiérték-feladat megoldását rajzolja az ábrába. Legelőször betölti az <code>options.mat</code> fájlba mentett struktúrát, amiben a szükséges adatokat tároljuk, majd a szükséges adatkiolvasás/hibaelhárítás után lekérdezzük, hogy melyik pixelre kattintottunk. Ezzel megkapjuk a kezdeti feltételt. A <code>numODE</code> függvény kimeneteként a (4.2) rendszert nyerjük <i>function handle</i> formában. Ennek előnye, hogy így nem kell minden változtatás után kézzel módosítani egy külső függvényt, ami a differenciálegyenlet-rendszert adja meg. A következő utasítással azt döntjük el, hogy a beolvasott struktúra <code>Abort</code> mezője milyen értékű. Ha 0, akkor az azt jelenti, hogy nincs időkorlát a numerikus megoldásra. Ha 1, akkor legfeljebb 5 másodpercig dolgozhat a megoldó a kezdetiérték-feladaton. Ezután a megoldó típusát választjuk ki. Az <code>munlock</code> és <code>clear</code> beépített függvényekkel a memóriából töröljük a hívó függvényt, ugyanis anélkül folyamatosan halmozódna az eltelt idő és így rövidesen nem tudnánk célunk szerint használni az időzítőt. Végül a megoldás ábrázolása következik a kívánt stílusok és színek alkalmazásával.
665-686	A <code>File</code> $\rightarrow$ <code>Save as...</code> lenyomására aktiválódik. Megnyitunk egy párbeszédablakot, ahol megadhatjuk a mentett kép elérési útját. Beépítettünk egy hibakezelő részt, ami akkor lép életbe, ha olyan helyre akarjuk menteni az ábrát, ahol nincs írási engedélyünk. Egy ilyen esetet mutat a 9 ábra.
688-691	A floppy ikonra kattintva a <code>saveDefault</code> függvény a MATLAB éppen használt könyvtárába menti el a képet <code>.bmp</code> formátumban <code>CurrentFigure</code> néven.

693-718	Gyors használati utasítást jelenít meg egy ablakban (lásd 8. ábra).
720-730	Információ a fejlesztőről.
733-736	A (4.2) differenciálegyenlet-rendszerből az $x' = 0$, $y' = 0$ alkalmazásával kapott algebrai egyenletrendszerre próbál egzakt megoldást találni.
738-744	A Jacobi-mátrixot határozzuk meg a definíciós összefüggésből.
747-774	A <code>dirfield</code> három bemenő paramétert vár. Az első kettő egy-egy vektor, ami a tartományt alkotó rácsot adja meg. A harmadik paraméter egy <i>function handle</i> , amely a dy/dx függvényt jellemzi. A <i>meshgrid</i> -del létrehozunk a két vektorból egy-egy mátrixot, ami a generált derékszögű hálót alkotja. Vesszük a harmadik paraméterként megadott kétváltozós függvény helyettesítési értékét az említett két mátrixban. Így rendelkezésünkre állnak a függvényértékek a csomópontokban. ² A vonalelemeket a <i>quiver</i> függvénnyel rajzoljuk a képernyőre, de előbb normalizáljuk a szakaszok hosszát (euklideszi normát véve), hogy egyenlő hosszúak legyenek, ne pedig a vektormező meredekségét jelezzék. Ennek ábrázolásbeli szerepe van.
776-805	A <code>numODE</code> feladata az, hogy az $f(x, y) =$ és $g(x, y) =$ melletti mezőkben megadott sztringekből egy <i>function handle</i> -t készítsen, ami a kezdetiérték-feladat numerikus megoldásához szükséges. A szorzás, osztás és a hatványozás elemenkénti megfelelőjét használjuk, továbbá a $z(1) = x$, $z(2) = y$ jelöléssel elérjük, hogy a MATLAB vektoregyenletként értelmezze a két skalár egyenletet.
807-837	A <code>stopInt</code> függvény a biztonság kedvéért lett beépítve. Működése a következő. A MATLAB beépített megoldói a beállításokat egy struktúra megadásával fogadják. A struktúra az <i>odeset</i> -tel adható meg. A struktúra egyik mezője az <i>OutputFcn</i> . Ha ehhez a tulajdonsághoz hozzárendelünk egy függvényt, akkor a megoldás során minden egyes sikeresen végrehajtott integrálási lépés során a numerikus megoldó végrehajtja a függvényt. A célunk az, hogy egy integrálgörbe meghatározásával ne töltsön 5 másodpercnél időt a program, így nem fog komolyabban lelassulni. Három változót – korábbi időpontot, jelenlegi időpontot és az eltelt időt – <i>persistent</i> típussal deklarálunk és minden hívás során az eltelt időt úgy kapjuk meg, hogy a jelenleg leolvasott időből kivonjuk az eggyel azelőtti függvényhívás idejét. Ezek összegzésével az összes eltelt időt mérjük. Amikor ez meghaladja az 5 másodpercet, a megoldó leáll és visszatér az addig kiszámított közelítő értékekkel.
840-867	Az egyes verziók változtatásai.

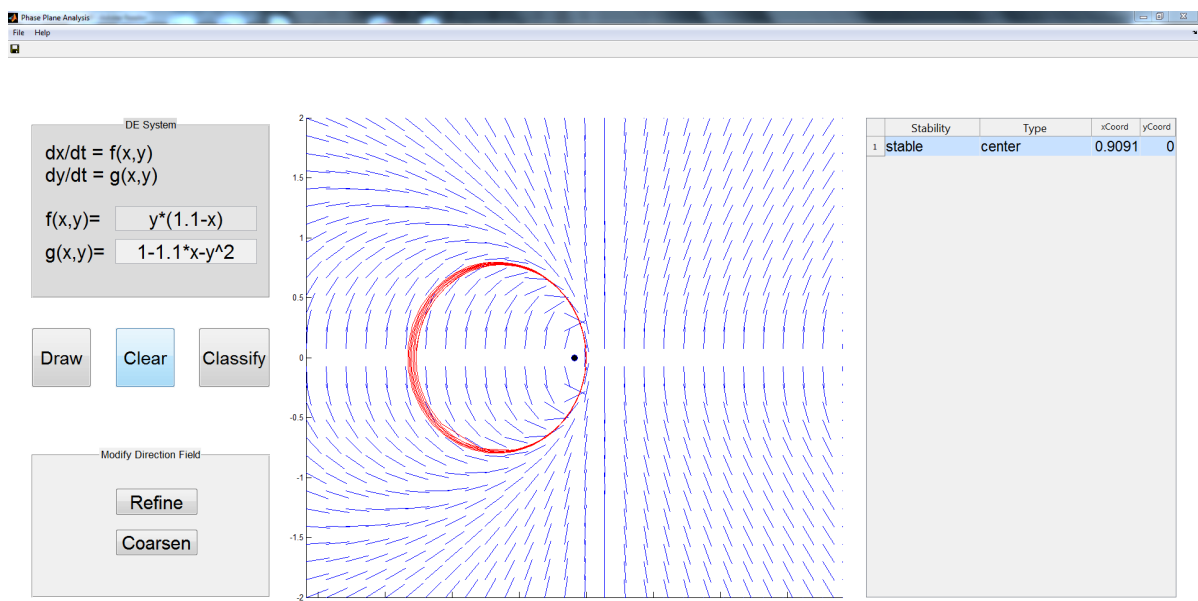
1. táblázat: A PhasePlaneAnalysis program leírása

Tekintsük a 4. fejezet 5. pontjában kézzel vázolt fázisportrékat és hasonlítsuk össze a programmal készített ábrákkal.

²Meg lehetne oldani kettős for-ciklusal is, de a MATLAB egy interpreted nyelv, így gyorsabb ha vektorizálunk.



10. ábra. A fázisportré $\alpha = 2$, $\beta = 1$, $q = 1$ esetén

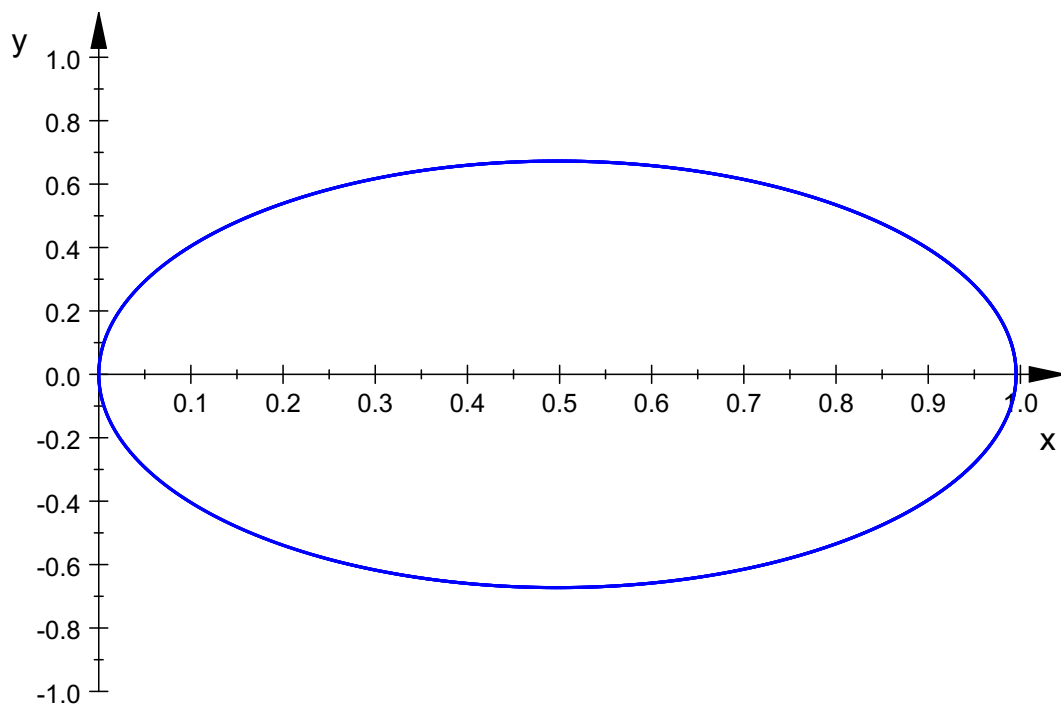


11. ábra. A fázisportré $\alpha = 1$, $\beta = 1$, $q = 1,1$ esetén

A 10. ábrán a paraméterek egy olyan variációját vettük, amikor $\alpha\beta - q^2 < 0$. Ugyanazt a jelleget kaptuk, mint ami a kézzel vázolt 3. ábrán látható. A 11. ábrán azt az esetet mutatjuk, amikor $\alpha\beta - q^2 < 0$. Ahogy már írtuk a 4. fejezet 3.(c) pontjában, itt csak egy kritikus pont van (P_1). Az ábrába egyszer kattintottunk, azaz egy kezdeti feltételt adtunk meg (köribelül a $(0,0)$ pontot). Úgy tűnik a képről, hogy határciklust kaptunk. Viszont ez nem lehetséges a 3. Tétel értelmében, továbbá a kritikus pont osztályozása is azt mutatja, hogy itt stabil centrumról van szó. Azaz az integrálgörbék periodikus megoldásra utalnak [16]. Ha felhasználjuk a (3.15) egzakt megoldást, akkor láthatjuk a Mupad³-del készített 12. ábrán, hogy valóban centrumot

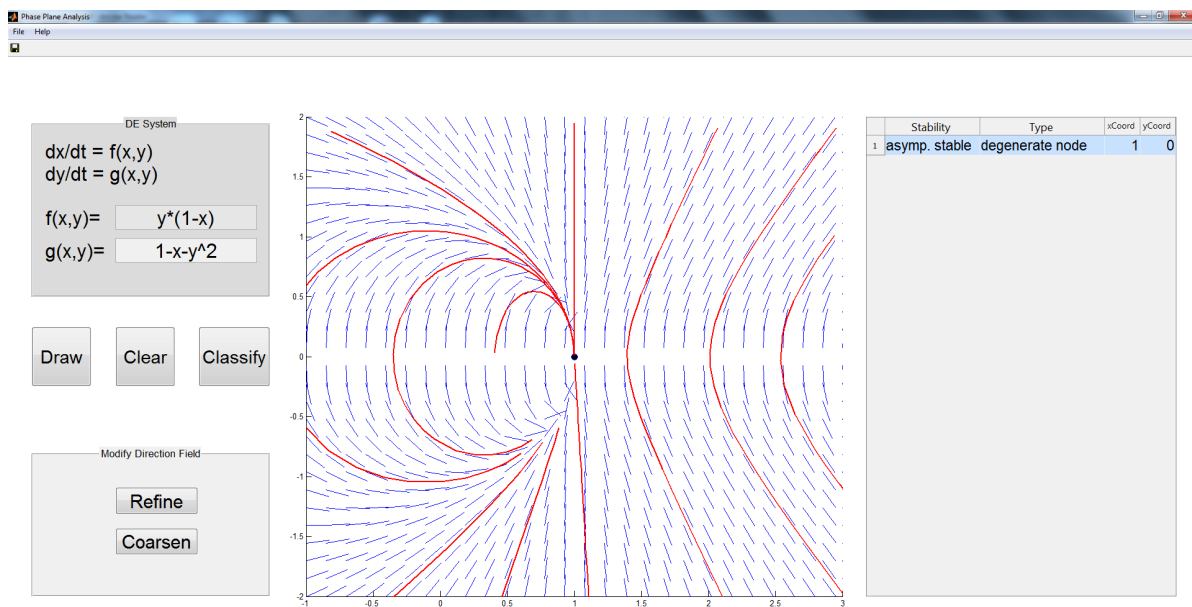
³A Mupad a Mathworks® része.

kapunk. Ez is mutatja az egzakt megoldás fontosságát és a numerikus megoldások esetenkénti tökéletlenségét.

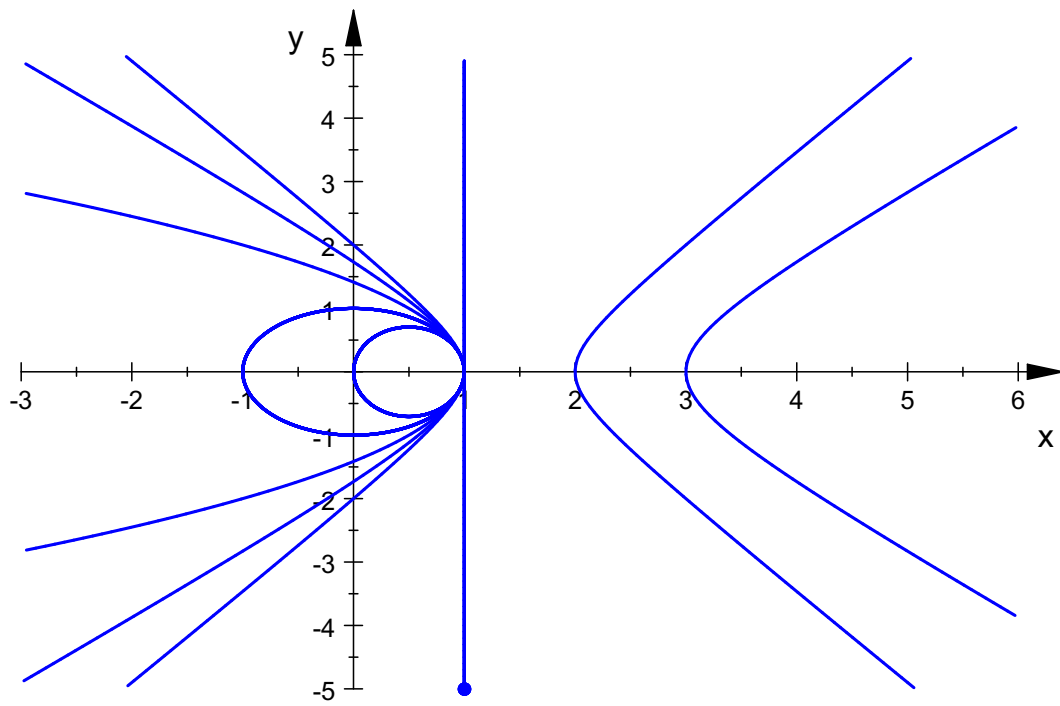


12. ábra. Az egzakt megoldás az $(x_0, y_0) = (0, 0)$ kezdeti feltétellel $\alpha = 1$, $\beta = 1$, $q = 1, 1$ esetén

Végül tekintsük az $\alpha\beta - q^2 = 0$ paraméterek közötti összefüggést. A program által készített iránymező a 13. ábrán, az egzakt megoldásból nyert görbék a 14. ábrán találhatók.



13. ábra. A fázisportré $\alpha = 1$, $\beta = 1$, $q = 1$ esetén



14. ábra. Az egzakt megoldás különböző kezdeti feltételek mellett $\alpha = 1$, $\beta = 1$, $q = 1$ esetén

Ebben az esetben nem volt probléma a numerikus megoldás során.

6. Összefoglalás, további cél

A vezikulum egy zárt foszfolipid membrán, amely a belsejében lévő folyadékot elválasztja a membránon kívül lévő folyadéktól. A lipid hárták biológiai jelentősége miatt egyre növekszik a fizikai érdeklődés a vezikulumok iránt. Viszkózus áramlásban elhelyezett vezikulumok dinamikájának tanulmányozása különösen jelentős, mert más – bonyolultabb viselkedésű – sejtek, mint például a vörösvérsejtek viselkedése megérthető ezen keresztül. A vezikulumoknak több egyensúlyi helyzete van, vizsgálatuk a nemlinearitás miatt bonyolult.

A dolgozatban vezikulumok mozgását leíró autonóm differenciálegyenlet-rendszer vizsgálatával foglalkoztunk. Egyes paraméter értékekre egzakt megoldást kerestünk, majd bizonyítottuk a megoldások létezését és egyértelműségét. A differenciálegyenlet-rendszer kvalitatív vizsgálatára lineáris stabilitásvizsgálatot végeztünk a fázissíkon, amelyből megmutatható, hogy milyen esetekben létezik periodikus megoldás, továbbá általános megoldást adtunk bármely paraméterre.

A fázisgörbék megjelenítésére egy – a Függelékben látható – saját készítésű általános célú MATLAB programot adtunk, amely – egyéb funkciókon kívül – kirajzolja az iránymezőt, a sajátértékek alapján elvégzi az egyensúlyi pontok osztályozását és megállapítja a stabilitást. A program felhasználható oktatási célra is.

További cél, hogy elkészítsük a programnak egy olyan változatát, amelyhez nem szükséges a Symbolic Math Toolbox, lemondva így arról az előnyről, hogy automatikusan keressen kritikus pontokat. Ebben az esetben a MATLAB Compiler segítségével létrehozható egy MATLAB független grafikus felület, ami olyan gépeken is használható, amelyeken nincs MATLAB telepítve (elegendő letölteni az ingyenes MATLAB Compiler Runtime-ot).

Köszönetnyilvánítás: A kutatás az Európai Unió és Magyarország támogatásával, az Európai Szociális Alap társfinanszírozásával a TÁMOP 4.2.4.A/2-11-1-2012-0001 azonosító számú „Nemzeti Kiválóság Program – Hazai hallgatói, illetve kutatói személyi támogatást biztosító rendszer kidolgozása és működtetése konvergencia program” című kiemelt projekt keretei között valósult meg.

7. Függelék

7.1. A PhasePlaneAnalysis függvény

```
1 function PhasePlaneAnalysis(xCoord,yCoord)
2 %PHASEPLANEANALYSIS is a GUI tool for visualizing the direction field of
3 %two coupled first order nonlinear autonomous differential equations.
4 % PHASEPLANEANALYSIS without input uses the Symbolic Math Toolbox to
5 % obtain the critical points.
6 % PHASEPLANEANALYSIS(XCOORD,YCOORD) takes the provided x and y
7 % coordinates of the critical points instead of invoking the Symbolic
8 % Math Toolbox to compute the roots.
9 % There are no output arguments.
10 % In order to obtain usage, click on the Usage submenu within the Help
11 % menu and read the informative tooltips.
12 %
13 % See also: quiver
14
15 % Input data must be provided in the DE System panel. Provide the
16 % functions using the usual MATLAB conventions (point before a
17 % multiplicative operation sign is not needed). Output data (equilibrium
18 % points) are displayed on the right. The Draw pushbutton sketches the
19 % direction field, the Classify pushbutton classifies the calculated
20 % equilibrium points according to their behavior. If you want a new blank
21 % canvas press Clear. Within the Modify Direction Field panel there are
22 % two pushbuttons. With these buttons you can change the number of line
23 % elements displayed on the figure. Other setting are available through
24 % File -> Options
25
26
27
28 %% Check input
29 % Symbolic Math Toolbox is required, determine if we have
30 installedSMT = hasSMT;
31 if ~installedSMT
32     error('Symbolic Math Toolbox is needed for the program.');
```

```

56     if strcmp(v(i).Name,'Symbolic Math Toolbox');
57         answer = 1;
58     end
59 end
60 end
61
62
63 %% Initialize main figure and axes
64 screenSize = get(0,'ScreenSize');
65 S.mainWindow = figure;
66 % Menu bar is not necessary, prevent the user draw into the figure
67 set(S.mainWindow, 'Resize','on', 'MenuBar','none', 'NumberTitle','off', ...
68     'Name','Phase Plane Analysis', 'Units', 'pixels', 'Color', 'white', ...
69     'Position',[1 1 0.6*screenSize(3) 0.6*screenSize(4)], ...
70     'Visible', 'off', 'DeleteFcn','close(''all'')');
71 movegui(S.mainWindow,'center');
72 set(gcf, 'InvertHardCopy', 'off', 'Visible', 'on');
73 S.canvas = axes;
74 % Set the axes invisible so that the interval of interest is available for
75 % function "initializeIVP"
76 set(S.canvas, 'Parent',S.mainWindow, 'HandleVisibility','callback', ...
77     'Units','normalized', 'XLim',[-1 1], 'Interruptible', 'off', ...
78     'YLim',[-1 1], 'Visible','off', ...
79     'Position',[0.25 0.1 0.45 screenSize(3)/screenSize(4)*0.45]);
80 if nargin == 2
81     xCoord = xCoord(:); % enforce xCoord and yCoord to be column vectors
82     yCoord = yCoord(:);
83     S.solution = [xCoord yCoord];
84 end
85
86 %% Create user interface controls for input
87 % Define background color
88 darkgrey = [0.86 0.86 0.86];
89 lightgrey = [0.9 0.9 0.9];
90 % Create control buttons, textboxes and static text fields
91 S.ODESystemPanel = uipanel('Parent',S.mainWindow, 'FontUnit', 'normalized',...
92     'Title','DE System', 'Units', 'normalized', 'Position',[0.02 0.6 0.2 0.3], ...
93     'TitlePosition','centertop', 'BackgroundColor',darkgrey);
94 systemString = sprintf('%s\n%s','dx/dt = f(x,y)','dy/dt = g(x,y)');
95 S.text_ODESystem = uicontrol('Parent',S.ODESystemPanel, 'Style','text', ...
96     'Units','normalized', 'FontUnits', 'normalized', 'FontSize',0.4, ...
97     'String',systemString, 'Position',[0.05 0.65 0.5 0.3], ...
98     'HorizontalAlignment', 'left', 'BackgroundColor',darkgrey, ...
99     'TooltipString',['<html><font size="5"> The program illustrates ', ...
100     'the phase portrait <br />of this <b>autonomous<b> system</font></html>']);
101 S.text_f = uicontrol('Parent',S.ODESystemPanel, 'Units','normalized', ...
102     'Style','text', 'String','f(x,y)=', 'FontUnits', 'normalized', ...
103     'Position',[0.05 0.4 0.3 0.15], 'HorizontalAlignment','left', ...
104     'BackgroundColor',darkgrey, 'FontSize',0.8);
105 S.text_g = uicontrol('Parent',S.ODESystemPanel, 'Units','normalized', ...
106     'Style','text', 'String','g(x,y)=', 'Position',[0.05 0.2 0.3 0.15], ...
107     'FontUnit','normalized', 'BackgroundColor',darkgrey, ...
108     'FontSize',0.8, 'HorizontalAlignment', 'left');
109 S.edit_f = uicontrol('Parent',S.ODESystemPanel, 'Units','normalized', ...
110     'Style','edit', 'String','Provide f(x,y)', 'FontUnit', 'normalized', ...
111     'FontSize',0.8, 'Position',[0.35 0.4 0.6 0.15], ...
112     'BackgroundColor',lightgrey, 'TooltipString', ...
113     ['<html><font size="5"> <b>Type</b> the right hand side of the ' ...
114     'first equation of the system</font><br />', ...
115     '<font size="5" color="red"><b>Note: </b></font><font size="5">', ...
116     'see the edit box below</font></html>']);

```

```

117 S.edit_g = uicontrol('Parent',S.ODESystemPanel, 'Units','normalized', ...
118     'Style','edit', 'String','Provide g(x,y)', 'FontUnit', 'normalized', ...
119     'FontSize',0.8, 'Position',[0.35 0.2 0.6 0.15], 'TooltipString', ...
120     ['<html><font size="5"> <b>Type</b> the right hand side of the ' ...
121     'second equation of the system</font><br />', ...
122     '<font size="5" color="red"><b>Note: </b></font><font size="5">', ...
123     'the callback belonging to this edit box performs<br />', ...
124     'the calculation, therefore you must press <b>Enter</b> ', ...
125     'here <br />every time the content of the above exit box changes', ...
126     '</font></html>']);
127
128 S.drawDirectionField = uicontrol('Parent',S.mainWindow, 'Style','pushbutton', ...
129     'String','Draw', 'Units','normalized', 'FontUnit','normalized', ...
130     'FontSize', 0.3, 'Position',[0.02 0.45 0.05 0.1], ...
131     'TooltipString',['<html><font size="5"> <b>Displays</b> the ', ...
132     'direction field</font></html>']);
133 S.clearScreen = uicontrol('Parent',S.mainWindow, 'Style','pushbutton', ...
134     'String','Clear', 'Units','normalized', 'Position',[0.09 0.45 0.05 0.1], ...
135     'FontUnit','normalized', 'FontSize',0.3, ...
136     'TooltipString',['<html><font size="5"> <b>Clears</b> the entire ', ...
137     'axes</font></html>']);
138 S.classify = uicontrol('Parent',S.mainWindow, 'Style','pushbutton', ...
139     'String','Classify', 'Units','normalized', 'FontUnit','normalized', ...
140     'FontSize',0.3, 'Position',[0.16 0.45 0.06 0.1], ...
141     'TooltipString',['<html><font size="5"> <b>Identifies</b> the '...
142     'type of critical points</font></html>']);
143 S.modifyDirectionField = uipanel('Parent',S.mainWindow, 'Units', 'normalized', ...
144     'Title','Modify Direction Field', 'Position',[0.02 0.1 0.2 0.25], ...
145     'FontUnit','normalized', 'TitlePosition','centertop');
146 S.refine = uicontrol('Parent',S.modifyDirectionField, 'Style','pushbutton', ...
147     'FontUnit', 'normalized', 'FontSize', 0.7, 'String','Refine', ...
148     'Units','normalized', 'Position',[0.35 0.6 0.35 0.2], ...
149     'TooltipString',['<html><font size="5"> <b>More</b> line elements', ...
150     ' are drawn</font></html>']);
151 S.coarsen = uicontrol('Parent',S.modifyDirectionField, 'Style','pushbutton', ...
152     'FontUnit', 'normalized', 'FontSize', 0.7, 'String','Coarsen', ...
153     'Units','normalized', 'Position',[0.35 0.3 0.35 0.2], ...
154     'TooltipString',['<html><font size="5"> <b>Less</b> line elements', ...
155     ' are drawn</font></html>']);
156
157 %% Create menus
158 S.FileMenu = uimenu('Parent',S.mainWindow, 'Label','File');
159     S.FileMenu_saveAs = uimenu('Parent',S.FileMenu, 'Label','Save as...');
160     S.FileMenu_Options = uimenu('Parent',S.FileMenu, 'Label','Options...');
161     uimenu('Parent',S.FileMenu, 'Label','Close', ...
162         'Callback','close(''all'')');
163 S.HelpMenu = uimenu('Parent',S.mainWindow, 'Label','Help');
164     S.HelpMenu_usage = uimenu('Parent',S.HelpMenu, 'Label','Usage', ...
165         'Callback',@howtouse);
166     S.HelpMenu_about = uimenu('Parent',S.HelpMenu, 'Label','About', ...
167         'Callback',@developer);
168
169 %% Create user interface objects for output
170 % Here the characteristics of the equilibrium points are displayed. The
171 % table is filled in with values when the Classify button is pressed.
172 columnWidth = cell(1,4);
173 columnWidth{1} = 0.08*screenSize(3);
174 columnWidth{2} = 0.105*screenSize(3);
175 columnWidth{3} = 0.03*screenSize(3);
176 columnWidth{4} = 0.03*screenSize(3);
177 lightblue = [200 225 255]/255; % RGB colors normalized

```

```

178 S.criticalPoints = uitable('Parent',S.mainWindow, 'Units', 'normalized',...
179     'Position',[0.72 0.1 0.26 0.8], ...
180     'ColumnName',{'<html><font size="5"> Stability </font></html>', ...
181         '<html><font size="5"> Type </font></html>', ...
182         '<html><font size="4"> xCoord </font></html>', ...
183         '<html><font size="4"> yCoord </font></html>'}, ...
184     'ColumnFormat',{'char','numeric','numeric'}, ...
185     'ColumnWidth', columnWidth, 'FontUnits','normalized', ...
186     'ColumnEditable',[false false false false], 'FontSize',0.03, ...
187     'BackgroundColor',lightblue, 'Visible','off');
188
189 %% Create save and load icons
190 S.toolbar = uitoolbar('Parent',S.mainWindow);
191 % This declaration is from the Product Help
192 S.saveFigure = uipushtool('Parent',S.toolbar, 'CData',...
193     iconRead(fullfile(matlabroot,'toolbox\matlab\icons\savedoc.mat')), ...
194     'TooltipString','<html><font size="5">Save the current figure</font></html>');
195
196 %% Create callbacks and default options
197 % For better perspicuity most callback functions are declared in a separate
198 % section
199 set(S.edit_f, 'Callback',{@obtain_f,S});
200 set(S.edit_g, 'Callback',{@obtain_g,S}, 'Enable','off');
201 set(S.drawDirectionField, 'Callback',{@sketchDirfield,S}, 'Enable','off');
202 set(S.clearScreen, 'Callback',{@restoreBlank,S});
203 set(S.classify, 'Callback',{@classifyPoints,S}, 'Enable','off');
204 set(S.refine, 'Callback',{@increaseDensity,S}, 'Enable','off');
205 set(S.coarsen, 'Callback',{@decreaseDensity,S}, 'Enable','off');
206 set(S.canvas, 'ButtonDownFcn',{@initializeIVP,S});
207 set(S.saveFigure, 'ClickedCallback',{@saveDefault});
208 set(S.FileMenu_saveAs, 'Callback',{@saveAs});
209 set(S.FileMenu_Options, 'Callback',{@options,S});
210
211 % Save the default settings for options
212 optionsStruct.timeSpan = 100;
213 optionsStruct.solverType = 'ode45';
214 optionsStruct.lineStyle = '-';
215 optionsStruct.lineWidth = 0.5;
216 optionsStruct.Color = 'red';
217 optionsStruct.Marker = 'none';
218 optionsStruct.Abort = 1;
219 % Save the structure as a .mat file that can be loaded from MATLAB R8 or later
220 save('options.mat', 'optionsStruct', '-v6');
221
222
223 end % main function ends here
224
225
226 %% Create Options submenu
227 function options(varargin)
228 % Open Options GUI where you can adjust several settings
229 S = varargin{3}; % get the structure
230 % Initialize main figure and axes
231 S.OptionsGUI = figure;
232 set(S.OptionsGUI, 'Resize','off', 'MenuBar','none', 'NumberTitle','off', ...
233     'Name','Options', 'Units', 'pixels', 'Color', 'white', 'Visible', 'off');
234 movegui(S.OptionsGUI,'center');
235 set(S.OptionsGUI, 'Visible', 'on');
236 % Define background color
237 darkgrey = [0.86 0.86 0.86];
238 lightgrey = [0.9 0.9 0.9];

```

```

239 % Create control buttons, textboxes and static text fields
240 S.OptionsSolverPanel = uipanel('Parent',S.OptionsGUI, 'Title','Solver', ...
241     'Units','normalized', 'Position',[0.05 0.6 0.9 0.3], ...
242     'TitlePosition','centertop', 'BackgroundColor',darkgrey);
243 S.text_SolverType = uicontrol('Parent',S.OptionsSolverPanel, ...
244     'Units','normalized', 'Style','text', 'String','Integrator', ...
245     'Position',[0.05 0.68 0.15 0.15],'HorizontalAlignment', 'left', ...
246     'FontSize',10, 'BackgroundColor',darkgrey, 'TooltipString', ...
247     '<html><font size="5"> Select the numerical solver </font></html>');
248 integratorString = {'ode45';'ode23';'ode113';'ode15s';'ode23s';'ode23t'};
249 S.popup_SolverType = uicontrol('Parent',S.OptionsSolverPanel, ...
250     'Units','normalized', 'Style','popupmenu', 'String',integratorString, ...
251     'Position',[0.2 0.7 0.15 0.15],'HorizontalAlignment', 'left', ...
252     'FontSize',10, 'BackgroundColor',darkgrey);
253 timespanString = {'Timespan';'[0,T], T='};
254 S.text_Timespan = uicontrol('Parent',S.OptionsSolverPanel, ...
255     'Units','normalized', 'Style','text', 'String',timespanString, ...
256     'Position',[0.05 0.2 0.15 0.3],'HorizontalAlignment', 'left', ...
257     'FontSize',10, 'BackgroundColor',darkgrey, 'TooltipString', ...
258     '<html><font size="5"> Set the integration timespan </font></html>');
259 S.edit_Timespan = uicontrol('Parent',S.OptionsSolverPanel, ...
260     'Units','normalized', 'Style','edit', 'String','100', ...
261     'Position',[0.2 0.2 0.15 0.15],'HorizontalAlignment', 'center', ...
262     'FontSize',10, 'BackgroundColor',lightgrey);
263 S.text_Abort = uicontrol('Parent',S.OptionsSolverPanel, ...
264     'Style','text', 'String','Stop integration after 5 seconds', ...
265     'Units','normalized', 'Position',[0.5 0.68 0.4 0.15], 'FontSize',10,...
266     'HorizontalAlignment', 'left', 'BackgroundColor',darkgrey, ...
267     'TooltipString',['<html><font size="5">Stop the solution process ', ...
268     'after 5 seconds <br /> and return the solution calculated so far</font></html>']);
269 S.check_Abort = uicontrol('Parent',S.OptionsSolverPanel, ...
270     'Style','checkbox', 'Units','normalized', 'Value',1, ...
271     'Position',[0.92 0.68 0.03 0.15], 'BackgroundColor',darkgrey);
272
273 S.OptionsLinePanel = uipanel('Parent',S.OptionsGUI, 'Title','Line Style', ...
274     'Units','normalized', 'Position',[0.05 0.2 0.9 0.3], ...
275     'TitlePosition','centertop', 'BackgroundColor',darkgrey);
276 S.text_LineStyle = uicontrol('Parent',S.OptionsLinePanel, ...
277     'Units','normalized', 'Style','text', 'String','LineStyle', ...
278     'Position',[0.05 0.68 0.15 0.15],'HorizontalAlignment', 'left', ...
279     'FontSize',10, 'BackgroundColor',darkgrey);
280 lineStyleString = {'solid';'dash';'dot';'dash-dot';'none'};
281 S.popup_LineStyle = uicontrol('Parent',S.OptionsLinePanel, ...
282     'Units','normalized', 'Style','popupmenu', 'String',lineStyleString, ...
283     'Position',[0.2 0.7 0.15 0.15],'HorizontalAlignment', 'left', ...
284     'FontSize',10, 'BackgroundColor',darkgrey);
285 S.text_LineWidth = uicontrol('Parent',S.OptionsLinePanel, ...
286     'Units','normalized', 'Style','text', 'String','LineWidth', ...
287     'Position',[0.05 0.18 0.15 0.15],'HorizontalAlignment', 'left', ...
288     'FontSize',10, 'BackgroundColor',darkgrey);
289 lineWidthString = {'0.5';'2';'3';'5'};
290 S.popup_LineWidth = uicontrol('Parent',S.OptionsLinePanel, ...
291     'Units','normalized', 'Style','popupmenu', 'String',lineWidthString, ...
292     'Position',[0.2 0.2 0.15 0.15],'HorizontalAlignment', 'left', ...
293     'FontSize',10, 'BackgroundColor',darkgrey);
294 S.text_Color = uicontrol('Parent',S.OptionsLinePanel, ...
295     'Units','normalized', 'Style','text', 'String','Color', ...
296     'Position',[0.69 0.68 0.11 0.15],'HorizontalAlignment', 'left', ...
297     'FontSize',10, 'BackgroundColor',darkgrey);
298 colorString = {'red';'green';'blue';'magenta';'cyan';'black'};
299 S.popup_Color = uicontrol('Parent',S.OptionsLinePanel, ...

```

```

300     'Units','normalized', 'Style','popupmenu', 'String',colorString, ...
301     'Position',[0.8 0.7 0.15 0.15],'HorizontalAlignment', 'left', ...
302     'FontSize',10, 'BackgroundColor',darkgrey);
303 S.text_Marker = uicontrol('Parent',S.OptionsLinePanel, ...
304     'Units','normalized', 'Style','text', 'String','Marker', ...
305     'Position',[0.69 0.18 0.11 0.15],'HorizontalAlignment', 'left', ...
306     'FontSize',10, 'BackgroundColor',darkgrey);
307 markerString = {'none';'+','o';'*';',';','x';'s';'d';'p';'h'};
308 S.popup_Marker = uicontrol('Parent',S.OptionsLinePanel, ...
309     'Units','normalized', 'Style','popupmenu', 'String',markerString, ...
310     'Position',[0.8 0.2 0.15 0.15],'HorizontalAlignment', 'left', ...
311     'FontSize',10, 'BackgroundColor',darkgrey);
312
313 S.saveSettings = uicontrol('Parent',S.OptionsGUI, 'Style','pushbutton', ...
314     'Units','normalized', 'Position', [0.85 0.1 0.1 0.05], ...
315     'String','Save', 'Callback',{@saveOptions,S}, 'TooltipString', ...
316     '<html><font size="5"> Save settings </font></html>');
317 end
318
319 function saveOptions(varargin)
320 S = varargin{3}; % get the structure
321 optionsStruct.timeSpan = str2double(get(S.edit_Timespan, 'String'));
322 value = get(S.popup_SolverType, 'Value');
323 integratorString = {'ode45';'ode23';'ode113';'ode15s';'ode23s';'ode23t'};
324 optionsStruct.solverType = integratorString{value};
325 value = get(S.popup_LineStyle, 'Value');
326 lineStyleString = {'-';'--';':';'-.'; 'none'};
327 optionsStruct.lineStyle = lineStyleString{value};
328 value = get(S.popup_LineWidth, 'Value');
329 lineWidth = {0.5;2;3;5};
330 optionsStruct.lineWidth = lineWidth{value};
331 value = get(S.popup_Color, 'Value');
332 colorString = {'red';'green';'blue';'magenta';'cyan';'black'};
333 optionsStruct.Color = colorString{value};
334 value = get(S.popup_Marker, 'Value');
335 markerString = {'none';'+','o';'*';',';','x';'s';'d';'p';'h'};
336 optionsStruct.Marker = markerString{value};
337 optionsStruct.Abort = get(S.check_Abort, 'Value');
338 % Save the structure as a .mat file that can be loaded from MATLAB R8 or later
339 save('options.mat', 'optionsStruct', '-v6');
340 end
341
342 %% Define callback functions
343 function obtain_f(varargin)
344 S = varargin{3}; % get the structure
345 set(S.edit_g, 'Enable','on');
346 end
347
348 function obtain_g(varargin)
349 S = varargin{3}; % get the structure
350 eq_g = get(S.edit_g, 'String');
351 eq_f = get(S.edit_f, 'String');
352 % Both equations are available now, solve them symbolically
353 if isfield(S,'solution') % if the user provided the solution, use it
354     xSol = S.solution(:,1);
355     ySol = S.solution(:,2);
356     structure.xSol = xSol; structure.ySol = ySol;
357     lastwarn(''); % up to now there is no warning
358     structure.warning = lastwarn;
359 else
360     lastwarn(''); % reset lastwarn to indicate warning of "solveSystem"

```

```

361     solution = solveSystem(eq_f,eq_g);
362     % Obtain the warning message when the solve command fails to provide an
363     % explicit solution
364     if isempty(lastwarn)
365         xSol = solution.x; ySol = solution.y;
366         structure.warning = lastwarn;
367         structure.xSol = xSol; structure.ySol = ySol;
368     else
369         structure.warning = lastwarn;
370     end
371 end
372 set(S.edit_g, 'UserData',structure);
373 set([S.drawDirectionField, S.classify], 'Enable','on');
374 end
375
376 function pointType = classifyPoints(varargin)
377 % Calculate the Jacobian to linearize the system and classify the critical
378 % points
379 S = varargin{3}; % get the structure
380 data = get(S.edit_g, 'UserData');
381 eq_g = get(S.edit_g, 'String');
382 eq_f = get(S.edit_f, 'String');
383 Jacobian = computeJac(eq_f,eq_g);
384 if isempty(data.warning)
385     xSol = data.xSol; ySol = data.ySol;
386     nroSolution = numel(xSol);
387     syms x y;
388     J = zeros(2,2,nroSolution);
389     pointType = cell(nroSolution,4);
390     % Determine the Jacobian at the critical points
391     for k = 1:nroSolution
392         J(:, :,k) = subs(Jacobian,[x,y],[xSol(k) ySol(k)]);
393         eigenValues = eig(J(:, :,k));
394         % Perform the classification in view of the eigenvalues
395         pointType{k,3} = eval(xSol(k));
396         pointType{k,4} = eval(ySol(k));
397         % If the solver could find a solution then the Jacobian matrix can
398         % not be singular
399         if isequal(eigenValues(1),eigenValues(2)) % repeated eigenvalues
400             if isequal(J(:, :,k),diag(eigenValues))
401                 if eigenValues(1) > 0
402                     pointType{k,1} = 'unstable';
403                     pointType{k,2} = 'star node';
404                 else
405                     pointType{k,1} = 'asympt. stable';
406                     pointType{k,2} = 'star node';
407                 end
408             else
409                 if eigenValues(1) > 0
410                     pointType{k,1} = 'unstable';
411                     pointType{k,2} = 'degenerate node';
412                 else
413                     pointType{k,1} = 'asympt. stable';
414                     pointType{k,2} = 'degenerate node';
415                 end
416             end
417         else % distinct eigenvalues
418             if isreal(eigenValues)
419                 if eigenValues(1)*eigenValues(2) < 0 % opposite sign
420                     pointType{k,1} = 'unstable';
421                     pointType{k,2} = 'saddle';

```

```

422         else % same sign
423             if eigenValues(1) > 0
424                 pointType{k,1} = 'unstable';
425                 pointType{k,2} = 'source node';
426             else % negative eigenvalues
427                 pointType{k,1} = 'asympt. stable';
428                 pointType{k,2} = 'sink node';
429             end
430         end
431     else % complex eigenvalues
432         if real(eigenValues) == 0
433             pointType{k,1} = 'stable';
434             pointType{k,2} = 'center';
435         else % non-zero real part
436             if eigenValues(1) < 0
437                 pointType{k,1} = 'asympt. stable';
438                 pointType{k,2} = 'spiral sink';
439             else
440                 pointType{k,1} = 'unstable';
441                 pointType{k,2} = 'spiral source';
442             end
443         end
444     end
445 end
446 end
447 elseif isequal(Jacobian,zeros(2,2))
448     % If we provide f(x,y)=g(x,y)=0, the Symbolic Math Toolbox returns with
449     % a warning: "Could not extract individual solutions. Returning a MuPAD
450     % set object."
451     pointType{1,1} = 'unstable';
452     pointType{1,2} = 'everywhere fixed';
453     pointType{1,3} = '';
454     pointType{1,4} = '';
455 elseif strcmp(data.warning(1:13), 'The solutions')
456     % Assume that "Warning: The solutions are parametrized by the symbols:"
457     % warning message occurred during the solution of a linear system,
458     % which means that the Jacobian matrix is the coefficient matrix and is
459     % singular.
460     if ~isempty(get(S.drawDirectionField, 'UserData'))
461         structure = get(S.drawDirectionField, 'UserData');
462         structure.warning = true;
463         set(S.drawDirectionField, 'UserData',structure)
464     end
465     eigenValues = eig(Jacobian);
466     pointType{1,2} = '';
467     pointType{1,3} = '';
468     if isequal(eigenValues,zeros(2,1)) % both eigenvalues are 0
469         pointType{1,1} = 'unstable';
470     else % only 1 eigenvalue is 0
471         if eigenValues(eigenValues~=0) < 0 % extract the non-zero eigenvalue
472             pointType{1,1} = 'stable';
473         else % the other eigenvalue is positive
474             pointType{1,1} = 'unstable';
475         end
476     end
477     pointType{1,2} = 'line of equilibria';
478 else
479     % Other type of warning
480     warningstring = {'A warning occurred during the symbolic solution';...
481                     'of the system. Perhaps the Maple kernel could';...
482                     'not find an explicit solution. Classification '; ...

```

```

483         'can not be performed.'});
484     warndlg(warningstring);
485     pointType = {'','Warning occurred','',''};
486 end
487 set(S.criticalPoints, 'Visible','on', 'Data',pointType);
488 end
489
490 function sketchDirfield(varargin)
491 [handle S] = deal(varargin{[1;3]}); % get the calling handle and structure
492 % Check if a former direction field is on screen (important if we modified
493 % the density because failing this, the two fields would overlap each other)
494 formerData = get(S.drawDirectionField, 'UserData');
495 if ~isempty(formerData) % check if formerData exists or not
496     if ishandle(formerData.quiverHandle) % check if quiverHandle exists or
497         delete(formerData.quiverHandle); % not (important for the first
498     end % button press when quiverHandle does not exist, so there is
499         % nothing to delete
500 end
501 equilibriumPoints = get(S.edit_g, 'UserData');
502 if ~isempty(equilibriumPoints.warning)
503     x = -2:0.2:2;
504     y = -2:0.2:2;
505     % Do not enable the density controls because they would not provide
506     % more information about the phase portrait.
507     set([S.refine, S.coarsen], 'Enable','off');
508 else
509     xSol = eval(equilibriumPoints.xSol);
510     ySol = eval(equilibriumPoints.ySol);
511     structure.plotCriticalPoints = line(xSol,ySol);
512     set(structure.plotCriticalPoints, 'Marker','o', 'MarkerSize',8, ...
513         'MarkerFaceColor', 'black', 'LineStyle','none'); hold on;
514     nInterval = 20;
515     xInterval = [min(xSol)-2 max(xSol)+2];
516     yInterval = [min(ySol)-2 max(ySol)+2];
517     structure.xInterval = xInterval;
518     structure.yInterval = yInterval;
519     x = linspace(xInterval(1),xInterval(2),nInterval);
520     y = linspace(yInterval(1),yInterval(2),nInterval);
521     structure.nInterval = nInterval;
522     set(S.canvas, 'XLim',xInterval, 'YLim',yInterval, ...
523         'XLimMode','manual', 'YLimMode','manual');
524 end
525 % With the use of "matlabFunction" the user does not need to rewrite the
526 % function to be plotted by hand
527 eq_g = get(S.edit_g, 'String');
528 eq_f = get(S.edit_f, 'String');
529 equation = matlabFunction(sym(eq_g)/sym(eq_f));
530 % If a constant function is returned, the function handle does not contain
531 % the independent variables. However the "dirfield" function needs it.
532 if isempty(strfind(func2str(equation), ',')) % search for a , representing
533     equation = func2str(equation); % (x,y)
534     closePar = strfind(equation,')'); % find ")" in the string
535     equation = horzcat('@(x,y)',equation(closePar+1:end));
536     equation = str2func(equation);
537 end
538 % A handle to the quiver plot is needed so that it can be deleted when a
539 % new quiver plot is displayed. An other method would be: set(S.canvas,
540 % 'NextPlot','replacechildren') and set the critical points and the
541 % integral curves to 'HandleVisibility', 'off'
542 structure.quiverHandle = dirfield(x,y,equation);
543 % FIXME: after this line sometimes get(S.canvas, ButtonDownFcn) = []

```

```

544 set(structure.quiverHandle, 'Color','blue', 'AutoScaleFactor',0.9);
545 set(handle,'UserData',structure);
546 if isempty(equilibriumPoints.warning)
547     % Since the phase portrait is drawn, enable the density controls
548     set([S.refine, S.coarsen], 'Enable','on');
549 end
550 % Set the axes visible because now the interval of interest is available
551 % for function "initializeIVP"
552 set(S.canvas, 'Visible','on');
553 end
554
555 function restoreBlank(varargin)
556 % Clears axes and make Refine and Coarsen buttons unclickable.
557 S = varargin{3}; % get the structure
558 cla;
559 set([S.refine, S.coarsen], 'Enable','off');
560 end
561
562 function increaseDensity(varargin)
563 % Functions "increaseDensity" and "decreaseDensity" get the necessary data
564 % from a structure stored in the UserData field of "sketchDirfield"
565 S = varargin{3}; % get the structure
566 dataFromdrawDirectionField = get(S.drawDirectionField, 'UserData');
567 xInterval = dataFromdrawDirectionField.xInterval;
568 yInterval = dataFromdrawDirectionField.yInterval;
569 nInterval = dataFromdrawDirectionField.nInterval + 2;
570 delete(dataFromdrawDirectionField.quiverHandle);
571 x = linspace(xInterval(1),xInterval(2),nInterval);
572 y = linspace(yInterval(1),yInterval(2),nInterval);
573 structure.nInterval = nInterval;
574 % Since the nInterval field of the structure changes, we must overwrite the
575 % elements of UserData, therefore the other - unchanged - data must also
576 % be adjusted
577 structure.xInterval = xInterval;
578 structure.yInterval = yInterval;
579 eq_g = get(S.edit_g, 'String');
580 eq_f = get(S.edit_f, 'String');
581 equation = matlabFunction(sym(eq_g)/sym(eq_f));
582 structure.quiverHandle = dirfield(x,y,equation);
583 set(structure.quiverHandle, 'Color','blue', 'AutoScaleFactor',0.9);
584 set(S.drawDirectionField,'UserData',structure);
585 end
586
587 function decreaseDensity(varargin)
588 % Similar to "increaseDensity".
589 S = varargin{3}; % get the structure
590 dataFromdrawDirectionField = get(S.drawDirectionField, 'UserData');
591 xInterval = dataFromdrawDirectionField.xInterval;
592 yInterval = dataFromdrawDirectionField.yInterval;
593 nInterval = dataFromdrawDirectionField.nInterval - 2;
594 delete(dataFromdrawDirectionField.quiverHandle);
595 x = linspace(xInterval(1),xInterval(2),nInterval);
596 y = linspace(yInterval(1),yInterval(2),nInterval);
597 structure.xInterval = xInterval;
598 structure.yInterval = yInterval;
599 structure.nInterval = nInterval;
600 eq_g = get(S.edit_g, 'String');
601 eq_f = get(S.edit_f, 'String');
602 equation = matlabFunction(sym(eq_g)/sym(eq_f));
603 structure.quiverHandle = dirfield(x,y,equation);
604 set(structure.quiverHandle, 'Color','blue', 'AutoScaleFactor',0.9);

```

```

605 set(S.drawDirectionField, 'UserData', structure);
606 end
607
608 function initializeIVP(varargin)
609 % Solve and display the associated IVP numerically with the selected solver
610 S = varargin{3}; % get the structure
611 load('options.mat'); % load data from Options menu
612 eq_f = get(S.edit_f, 'String');
613 eq_g = get(S.edit_g, 'String');
614 dataFromdrawDirectionField = get(S.drawDirectionField, 'UserData');
615 try
616     xInterval = dataFromdrawDirectionField.xInterval;
617     yInterval = dataFromdrawDirectionField.yInterval;
618 catch
619     % If xInterval does not exist (since the solve command returned an
620     % error), use this predefined domain
621     xInterval = [-2 2];
622     yInterval = [-2 2];
623 end
624 set(S.canvas, 'XLim', xInterval, 'YLim', yInterval, 'XLimMode', 'manual', ...
625     'YLimMode', 'manual');
626 selectedPoint = get(S.canvas, 'CurrentPoint');
627 fHandle = numODE(eq_f, eq_g);
628 switch optionsStruct.Abort
629     case 0 % no time limit for numeric integration
630         opt = []; % use default settings
631     case 1 % time limit for numeric integration
632         opt = odeset('OutputFcn', @stopInt);
633 end
634 switch optionsStruct.solverType
635     case 'ode45'
636         % We do not need the argument "t" however MATLAB introduced the
637         % ability to use the tilde operator instead of an argument name in
638         % R2009b, so we retain "t" for a backward compatibility issue.
639         [t x] = ode45(fHandle, [0 optionsStruct.timeSpan], ...
640             [selectedPoint(1,1) selectedPoint(1,2)], opt); %#ok<*ASGLU>
641     case 'ode23'
642         [t x] = ode23(fHandle, [0 optionsStruct.timeSpan], ...
643             [selectedPoint(1,1) selectedPoint(1,2)], opt);
644     case 'ode113'
645         [t x] = ode113(fHandle, [0 optionsStruct.timeSpan], ...
646             [selectedPoint(1,1) selectedPoint(1,2)], opt);
647     case 'ode15s'
648         [t x] = ode15s(fHandle, [0 optionsStruct.timeSpan], ...
649             [selectedPoint(1,1) selectedPoint(1,2)], opt);
650     case 'ode23s'
651         [t x] = ode23s(fHandle, [0 optionsStruct.timeSpan], ...
652             [selectedPoint(1,1) selectedPoint(1,2)], opt);
653     case 'ode23t'
654         [t x] = ode23t(fHandle, [0 optionsStruct.timeSpan], ...
655             [selectedPoint(1,1) selectedPoint(1,2)], opt);
656 end
657 munlock('PhasePlaneAnalysis'); % allow clearing function from memory
658 clear('PhasePlaneAnalysis'); % clear function from memory
659 % Plot the y(x) function and keep the former ones
660 hold on; line(x(:,1), x(:,2), 'Marker', optionsStruct.Marker, ...
661     'LineStyle', optionsStruct.lineStyle, ...
662     'Color', optionsStruct.Color, 'LineWidth', optionsStruct.lineWidth);
663 end
664
665 function saveAs(varargin)

```

```

666 prompt = 'Filename (extension recommended)';
667 dialogTitle = 'Save';
668 numberOfLines = 1;
669 defaultAnswer = {'CurrentFigure.bmp'};
670 fileName = inputdlg(prompt, dialogTitle, numberOfLines, defaultAnswer);
671 if ~isempty(fileName)
672     try
673         set(0, 'DefaultFigureInvertHardcopy','off');
674         saveas(gcf,fileName{1}); % somehow it does not save the axes,
675                                 % it captures the whole figure
676     catch exception % if an error occurs
677         if (strcmp(exception.identifier,'MATLAB:saveas:invalidFilename'))
678             errorstring = sprintf('%s', ...
679                 'You have no writing permission into ', fileName{1});
680             errordlg(errorstring, 'Error during saving');
681         else
682             rethrow(exception);
683         end
684     end
685 end
686 end
687
688 function saveDefault(varargin)
689 % Saves figure to current directory with extension .bmp
690 saveas(gcf, 'CurrentFigure.bmp');
691 end
692
693 function howtouse(varargin)
694 % Quick user's guide
695 message = {'1. Write the function into the editbox near the text f(x,y)=';
696           ''
697           '2. Provide function g(x,y) in the same way';
698           ''
699           '3. Press button Draw';
700           ''
701           '4. Press button Classify';
702           ''
703           '5. Modify the density of arrows with Refine and Coarsen, if needed';
704           ''
705           '6. Click on the figure to set the initial values for the IVP';
706           'which is solved numerically';
707           ''
708           ''
709           'Step 3 and step 4 are interchangeable first time, then there';
710           'is no specific order at all.';
711           'To clear the graph, press Clear.'
712           'To save the figure, click on the save icon. To save it with';
713           'a different name, push File menu and then Save as...';
714           'Other settings can be given by selecting File -> Options...';
715           ''
716           'Hold your mouse over an entry to display tooltip help'};
717 msgbox(message, 'How to use');
718 end
719
720 function developer(varargin)
721 % About the creator and the terms of use
722 message = {'Phase Plane Analysis';
723           'Copyright 2013';
724           'Zoltán Csáti';
725           'University of Miskolc, Hungary';
726           ''};

```

```

727         'Use and distribute freely for non-commercial purposes,';
728         'but give appropriate credit to author Zoltán Csáti'};
729 msgbox(message, 'About Phase Plane Analysis');
730 end
731
732 %% Define subfunctions
733 function sol = solveSystem(f,g)
734 % Get the string and solve the system
735 sol = solve(horzcat(f,'=0'),horzcat(g,'=0'),'x','y');
736 end
737
738 function J = computeJac(f,g)
739 % Compute the Jacobian matrix to linearize the nonlinear system
740 f = sym(f);
741 g = sym(g);
742 J = [diff(f,'x') diff(f,'y');
743      diff(g,'x') diff(g,'y')];
744 end
745
746 %% Define functions for the numerical solution of the system of ODEs
747 function handle = dirfield(x,y,fun4dirfield)
748 % HANDLE = DIRFIELD(X,Y,EQUATION) creates the direction field at points
749 % determined by X and Y to the autonomous differential equation system
750 % provided by FUN4DIRFIELD and returns its handle HANDLE.
751
752 % Let us take the differential equation system y'=F(x,y) where function
753 % F:  $R \times R^n \rightarrow R^n$ . In the neighborhood of (x,y) the system can be
754 % described with a line element with slope y'=k where k is the quotient of
755 % the opposite(v) and the adjacent(u) sides. Let us choose u to 1. Then
756 % v=F(x,y). Here, u and v can be regarded as components of a vector and so
757 % we may use MATLAB's quiver function.
758 [Xmesh Ymesh] = meshgrid(x,y);
759 v = fun4dirfield(Xmesh,Ymesh);
760 % If function handle fun4dirfield is a scalar, evaluating it at Xmesh,Ymesh
761 % results in a scalar value. To have a field, this value must be
762 % concatenated as a mesh.
763 if isscalar(v)
764     v = v*ones(size(Xmesh));
765 end
766 u = ones(size(v));
767 % Set the length of the lines equal for better visualization
768 L = sqrt(u.^2 + v.^2); % L=norm(velocity,2)
769 % Return the handle of the quiver plot for later modifications
770 handle = quiver(Xmesh,Ymesh,u./L,v./L);
771 % Do not show the arrows since the do not represent the time dependency of
772 % the differential equation system because time is eliminated
773 set(handle, 'ShowArrowHead','off');
774 end
775
776 function handle = numODE(f,g)
777 %NUMODE Given strings f and g, NUMODE transforms them to a function handle.
778 % NUMODE is a function used for PhasePlaneAnalysis, it can only be used
779 % for two first order autonomous differential equations like
780 %  $x(t)' = f(x(t),y(t))$ , (1)
781 %  $y(t)' = g(x(t),y(t))$ . (2)
782 % It is much less flexible than function "odeToVectorField", however it
783 % does not need Symbolic Math Toolbox and work for older MATLAB versions.
784 %
785 % HANDLE = NUMODE(F,G) returns the required function handle HANDLE of the
786 % differential equation system (1)-(2) so that the user does not need to
787 % edit an external function when the right hand side is changed.

```

```

788
789 % Replace multiplication, division and power with their element-wise form
790 f = strrep(f, 'x*', 'x.*'); f = strrep(f, 'y*', 'y.*');
791 g = strrep(g, 'x*', 'x.*'); g = strrep(g, 'y*', 'y.*');
792 f = strrep(f, 'x/', 'x./'); f = strrep(f, 'y/', 'y./');
793 g = strrep(g, 'x/', 'x./'); g = strrep(g, 'y/', 'y./');
794 f = strrep(f, 'x^', 'x.^'); f = strrep(f, 'y^', 'y.^');
795 g = strrep(g, 'x^', 'x.^'); g = strrep(g, 'y^', 'y.^');
796 % Replace dependent variables x and y with the first and second element of
797 % a new vector, z
798 cellstring = strrep({f,g}, 'x', 'z(1)');
799 cellstring = strrep(cellstring, 'y', 'z(2)');
800 % Perceive the string expression which describes (1)-(2) as a vector
801 % differential equation
802 finalString = strcat('@(t,z)', '[', cellstring{1}, ','; cellstring{2}, ']');
803 % Convert the string to the required function handle
804 handle = str2func(finalString);
805 end
806
807 function status = stopInt(t,y,flag)
808 %STATUS = STOPINT(T,Y,FLAG) provides a stopping criterion with the required
809 % syntax for the OutputFcn property in odeSet.
810 % If the solution process takes more than 5 seconds, the solver
811 % automatically stops and returns the solution calculated to that time.
812 %
813 % See also: OutputFcn
814 timeBound = 5;
815 persistent elapsedTime currentTime_old currentTime_new
816 status = 0;
817
818 if isempty(elapsedTime)
819     time = clock;
820     minute = time(5);
821     second = time(6);
822     initTime = 60*minute + second;
823     elapsedTime = 0; currentTime_old = 0; currentTime_new = initTime;
824 else
825     time = clock;
826     minute = time(5);
827     second = time(6);
828     currentTime_old = currentTime_new;
829     currentTime_new = 60*minute + second;
830     elapsedTime = elapsedTime + currentTime_new - currentTime_old;
831 end
832 if elapsedTime >= timeBound
833     status = 1;
834     return
835 end
836
837 end
838
839
840 %% Changelog
841 %
842 % ver1_0: First release
843 %
844 % ver1_1: - Classification of critical points is done by using the
845 %         determinant-trace diagram instead of eigenvalues
846 %         - The user does not need to maintain an external function for the
847 %         numerical solution
848 %

```

```

849 % ver1_2: - Options menu created to provide different settings
850 %         - Error handling
851 %         - Solved the problem when the solve command returned a warning
852 %
853 % ver2_0: - Options menu amended with linestyle control and a security
854 %         option to stop integration during a long process
855 %         - Options menu need not to be open to use up altered settings
856 %         - Use .mat file instead of guidata to store and retrieve options
857 %         - No need for nested function -> backward compatibility
858 %         - Included HTML-based tooltips and updated help
859 %         - Instead of (hObject, eventdata, varargin), varargin is used in
860 %         callback functions
861 %         - Set fontsize to normalized, now fonts are enlarged when the
862 %         window is resized
863 %
864 % final: - Eigenvalue-based classification (again)
865 %         - New column in the output table
866 %         - Fixed grammar mistakes
867 %         - Fixed some bugs

```

7.2. Az iconRead függvény

```
1 function cdata = iconRead(filename,guessalpha)
2 % ICONREAD read an image file and convert it to CData for a HG icon.
3 %
4 % CDATA=ICONREAD(FILENAME)
5 %   Read an image file and convert it to CData with automatic transparency
6 %   handling. If the image has transparency data, PNG files sometimes do,
7 %   the transparency data is used. If the image has no CData, the top left
8 %   pixel is treated as the transparent color.
9 %
10 % CDATA=ICONREAD(FILENAME, FALSE)
11 %   Same as above but suppress the usage of the top left pixel for images
12 %   with no transparency data. This may require the caller to handle the
13 %   transparency explicitly. View the contents of this m-file for an
14 %   example of how to handle transparency.
15 %
16 % Example:
17 %
18 % icon = fullfile(matlabroot,'toolbox','matlab','icons','matlabicon.gif');
19 % uitoggletool('CData',iconread(icon));
20 %
21 % See also IMREAD.
22
23 % Copyright 1984-2007 The MathWorks, Inc.
24
25 if nargin < 2
26     guessalpha = true;
27 end
28
29 [p,f,ext] = fileparts(filename);
30 % if this is a mat-file, look for the variable cdata (or something like it)
31 if isequal(lower(ext),'.mat')
32     cdata = [];
33     s = whos('-file',filename);
34     for i=1:length(s)
35         if ~isempty(strfind(lower(s(i).name), 'cdata'))
36             data = load(filename,s(i).name);
37             cdata = data.(s(i).name);
38         end
39     end
40     return
41 end
42
43 [cdata,map,alpha] = imread(filename);
44 if isempty(cdata)
45     return;
46 end
47
48 if isempty(map)
49     if isinteger(cdata)
50         cname = class(cdata);
51         cdata=double(cdata);
52         cdata = cdata/double(intmax(cname));
53     else
54         cdata=double(cdata);
55         cdata = cdata/255;
56     end
57 else
```

```

58     cdata = ind2rgb(cdata,map);
59 end
60
61 if isempty(alpha)
62     if ~guessalpha
63         return;
64     end
65     % guess the alpha pixel by using the top left pixel in the icon
66     ap1 = cdata(1,1,1);
67     ap2 = cdata(1,1,2);
68     ap3 = cdata(1,1,3);
69     alpha = cdata(:, :, 1) == ap1 & cdata(:, :, 2) == ap2 & cdata(:, :, 3) == ap3;
70     alpha = ~alpha;
71 end
72
73 % process alpha data
74 r = cdata(:, :, 1);
75 r(alpha == 0) = NaN;
76 g = cdata(:, :, 2);
77 g(alpha == 0) = NaN;
78 b = cdata(:, :, 3);
79 b(alpha == 0) = NaN;
80 cdata = cat(3,r,g,b);

```

8. Hivatkozások

- [1] Brauer, F. and Nohel, J. A. *The Qualitative Theory of Ordinary Differential Equations: An Introduction*. Dover Publications, 1989.
- [2] Cox, R. G. The deformation of a drop in a general time-dependent fluid flow. *Journal of Fluid Mechanics*, 37:601–623, 1969.
- [3] Drochon, A. Rheology of dilute suspensions of red blood cells: experimental and theoretical approaches. *Eur. Phys. J. Appl. Phys*, 22:155–162, 2003.
- [4] Einstein, A. *Eine neue Bestimmung der Moleküldimensionen*. PhD thesis, Universität Zürich, Bern, 1905.
- [5] Farkas, M. *Speciális függvények*. Műszaki Könyvkiadó, 1964.
- [6] Frankel, N. A. and Acrivos, A. The constitutive equation for a dilute emulsion. *Journal of Fluid Mechanics*, 44:65–78, 1969.
- [7] Glendinning, P. *Stability, Instability and Chaos: An Introduction to the Theory of Nonlinear Differential Equations*. Cambridge University Press, 1994.
- [8] Guedda, M., Abaidi, M., Benlahsen, M., and Misbah, C. Dynamic modes of quasispherical vesicles: Exact analytical solutions. *Phys. Rev.*, 86:051915, 2012. doi: 10.1103/PhysRevE.86.051915.
- [9] Jeffrey, G. B. The motion of ellipsoidal particles immersed in a viscous fluid. *Proc. Roy. Soc. London*, 102:161–179, 1922.
- [10] Keller, S. R. and Skalak, R. Motion of a tank-treading ellipsoidal particle in a shear flow. *J. Fluid Mech.*, 120:27–47, 1982.
- [11] Khader, M. M. Computational Approaches for Solving the Logistic Differential Equation Using VIM-Padé and Chebyshev-Spectral Techniques. *IJNS*, 15(2):157–167, 2013.
- [12] Lamb, H. *Hydrodynamics*. Cambridge University Press, 1895.
- [13] Mader, M. A., Vitkova, V., Abkarian, M., Viallat, A., and Podgorski, T. Dynamics of viscous vesicles in shear flow. *Eur. Phys. J. E*, 19:389–397, 2006.
- [14] Misbah, C. Vacillating breathing and tumbling of vesicles under shear flow. *Phys. Rev. Lett.*, 96(2):028104, 2006.
- [15] Polyanin, A. D. and Zaitsev, V. F. *Handbook of Exact Solutions for Ordinary Differential Equations*. Chapman and Hall/CRC, second edition, 2002.
- [16] Rontó, M. and Raisz, P. *Differenciálegyenletek műszakiaknak*. Miskolci Egyetemi Kiadó, 2004.
- [17] Taylor, G. I. The motion of ellipsoidal particles in a viscous fluid. *Proc. Roy. Soc. London*, 103:58–61, 1923.

- [18] Taylor, G. I. The viscosity of a fluid containing small drops of another fluid. *Proc. Roy. Soc. London*, 138:41–48, 1932.
- [19] Vitkova, V., Mader, M. A., Biben, T., and Podgorski, T. Tumbling of lipid vesicles, enclosing a viscous fluid, under a shear flow. *Journal of Optoelectronics and Advanced Materials*, 7(1):261–264, 2005.
- [20] Vlahovska, P. M., Podgorski, T., and Misbah, C. Vesicles and red blood cells in flow: From individual dynamics to rheology. *C. R. Phys.*, 10:775–789, 2009.