Teknillinen korkeakoulu Apiola/Tikanmäki
Matematiikka

Mat-1.1332 Matematiikan peruskurssi KP3-II syksy 2008
`http://www.math.hut.fi/teaching/kp3-ii/08`

Laskuharjoitus 6 (viikko 50 , 8 – 12.12.2008)

Pääsivu: `http://math.tkk.fi/opetus/kp3-ii/08/`
Luentosivu: `http://math.tkk.fi/opetus/kp3-ii/08/L/Luento1-36.html`
Harjoitussivu: `http://math.tkk.fi/opetus/kp3-ii/08/H/`
Matlab: `http://math.tkk.fi/opetus/kp3-ii/08/matlab/`
Matlab ODE-ohje:
`http://math.tkk.fi/opetus/kp3-ii/08/matlab/ODE_ohje.html`

Muista: Jos käytät MATLAB:ia ssh-pääteyhteydellä, niin valitse sivulta `http://www.tkk.fi/atk/luokat/computernames.html` jokin ATK-luokan kone. Ja sitten: `use matlab`, jonka jälkeen käynnistys: `matlab`.
Huomaa, että yleiskoneissa `vipunen, kosh` ei MATLAB:ia ole.

Alkuviikko

1. Tarkastellaan yhtälöä $y' = -2\alpha(t-1)y$. Totea, että yleinen ratkaisu on: $y(t) = C_1\, e^{-\alpha\, t\,(t-2)}$

   (a) Muodosta tälle yhtälölle EM -askeleen kaava: $y_{n+1} = y_n + \ldots$

   (b) Muodosta BE-askeleen (implisiittinen Euler) kaava (ratkaistussa muodossa). Lineaarisen yhtälön tapauksessahan se on helppoa.

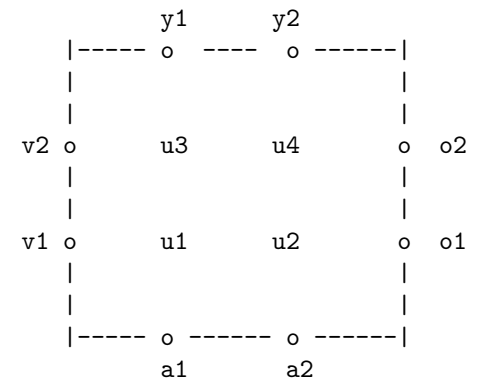
   Kts. EM ja BE kääntöpuolen kaavoista

2. (a) Millä t:n arvoilla edellisen tehtävän yhtälö on stabiili ja millä epästabiili? Piirrä ratkaisufunktioparven kuvaajia (käsinkin voit hahmotella, mutta MATLAB:ia tai vaikka graafista laskinta voit käyttää lisäapuna). Miten (epä)stabiilisuus ilmenee kuvassa.

   (c) Osoita, että BE:n tapauksessa (teht. 1 (b)) $y_n \to 0$, kun $n \to \infty$, valittiinpa $h > 0$ miten suureksi tahansa. Tämän ilmentää sitä, että BE ei aseta stabiilisuudelle mitään askelpituusrajoitusta, kunhan yhtälö on stabiili.

3. (a) Muodosta kuvan neliöalueella Laplacen yhtälön $\Delta u = 0$ differenssimenetelmäratkaisun yhtälösysteemi $Au = b$. Tämän ratkaisuvektori $u = [u_1, u_2, u_3, u_4]^T$ antaa siis approksimaatiot ratkaisufunktion arvoille näissä hila(sisä) pisteissä.

   Reuna-arvoina ovat vektorit o, y, v, a (oikea, ylä, vasen, ala).



   (b) Olkoon kuvan alue neliö $[0,3]\times[0,3]$ ja askel $h = 1$. Olkoot u:n reuna-arvot annettu näin: Vasen reuna: $u = 0$, alareuna: $u = x^3$, oikea reuna: $u = 27 - 9\,y^2$, yläreuna: $u = x^3 - 27\,x$.

   Määritä potentiaalifunktion (tai tasapainolämpötilan) $u(x, y)$ likiarvot sisäpisteissä $(1,1), (2,1), (1,2), (2,2)$.

   Anna vastaus kuvan mukaisena ruudukkona, jossa merkitset sisäsolmuihin ko. u:n arvot ja reunasolmuihin ao. reuna-arvot.

   Ohje: Saat halutessasi käyttää $4\times4-$ yhtälösysteemiin MATLAB:n "takakenoa": `>> u=A\b` Laskinavusteinen käsinlaskukaan ei liene toivottoman työläs.

   Vast: (ilman muotoilua) $\boldsymbol{u} = [-2 \;\; 2 \;\; -11 \;\; -16]^T$

Loppuviikko

1. Laske (käsin) kolme `RK4`-askelta tehtävälle $y' = t + y, \;\; y(0) = 1$, kun askel $h = 0.1$. Piirrä pisteet ja murtoviiva.

2. Kirjoita yksi askel Runge-Kutta menetelmää (RK4), yhtälölle $y' = ay, \;\; y(0) = 1$, askel h. Huomaa, että autonomisen yhtälön tapauksessa RK-kaavat voi kirjoittaa yhden muuttujan funktiolle $f(y)$, ts. kaavoissa esiintyvät $t-$ argumentit eivät tule käyttöön.

3. Totea, että O on systeemin $\begin{cases} x' = -\frac{3}{2}x - 10(1+x)y \\ y' = 5x - \frac{1}{2}y \end{cases}$ kriittinen piste ja linearisoi systeemi siinä.

   (a) Mikä on KRP:n tyyppi ja stabiilisuus? Riittää, kun lasket Jacobiaanin O:ssa ja `eig`:lla sen ominaisrvot.
   (b) Ratkaise sitten AA-tehtävä $x(0) = 1, y(0) = 1$ Eulerin menetelmällä käyttäen valmista funktiotamme tiedostossa (`../matlab/eulerV.m`). Ota askelpi-

tuudeksi $h = 0.1$ ja piirrä faasimurtoviivaa. Huomannet, että menetelmä käyttäytyy jopa kvalitatiivisesti väärin. Voit toki kokeilla pienemmällä askeleella.
(c) Suorita sama lasku samalla askeleella $h = 0.1$ käyttäen funktiotamme `rk4V`. Lisää piirros samaan kuvaan. (Suoritukset ovat teknisesti täsmälleen samanlaisia.)
(d) Laske ja piirrä lopuksi "tarkka approksimaatio" MATLAB:n `ode45`-funktiolla (adaptiivisella *Runge-Kutta-Fehlberg*- menetelmällä).
(Kts. `../matlab/ODE_ohje.html`.)

Sano jotain menetelmien vertailuksi (ellet mennyt sanattomaksi).

Suorituksesta: Kerro, miten olet toiminut, kirjoita pääkohdat koodeista, ja hahmottele kuvia taululle (tai näytä paperilla).
Sovitaan, että (d)-kohta on vapaaehtoinen.

4. Neliöalueessa diskretoidun Laplacen yhtälön matriisi on muotoa:

$$A = \begin{bmatrix} B & I & O & \dots & O \\ I & B & I & O & \dots \\ O & \ddots & \ddots & \ddots & O \\ \vdots & & & & \\ O & \dots & I & B & I \\ O & \dots & \dots & I & B \end{bmatrix}, \text{ missä } B = \begin{bmatrix} -4 & 1 & 0 & \dots & 0 \\ 1 & -4 & 1 & 0 & \dots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \\ 0 & \dots & 1 & -4 & 1 \\ 0 & \dots & \dots & 1 & -4 \end{bmatrix}$$

ja I on yksikkömatriisi sekä O nollamatriisi.

Jos solmuja on reunat mukaan lukien $n \times n$ kpl., niin sisäsolmuja on $(n-2)^2$ kpl. Tällöin B, I, O ovat $(n-2)\times(n-2)$ ja A on $(n-2)^2 \times (n-2)^2$ matriisi. A koostuu siis kolmesta lohkonauhasta, diagonaalilohkoina B ja ylä- ja alanauhalohkoina I.

Tällaisen matriisin muodostamiseen on monta tapaa. Kaikkein helpoin on valmiiksi ohjelmoitu MATLAB-funktio `delsq`. Koska matriisit ovat yleensä suuria, on syytä käsitellä niitä *harvoina* (engl. *sparse*), ts. nollia ei talleteta, ainoastaan nollasta poikkeavat indekseineen. Niinpä `delsq` rakentaa matriisista automaattisesti harvan. Jos haluat katsoa sitä tai sen osaa täytenä, käytä komentoa `full`. Kokeile:

```
help delsq
help numgrid
G=numgrid('S',10) % 'S' viittaa alueeseen ``Square''. Muitakin
A=delsq(G)        % on Matlabin numgrid-repertuaarissa.
full(A)
ans(1:10,1:10)
```

(Yhtä helppoa nelioalueen tapauksessa on käyttää omatekoista `lapm`-funktiota `matlab`-hakemistossa.)
Lopussa muita tapoja:

Kokeile näitä komentoja muutellen sopivasti parametreja. Suorita myös komento `help sparfun`, saadaksesi katsauksen tärkeään osaan MATLAB:ia: harvojen matriisien käsittelyfunktioihin.

Ratkaise tehtävä AV 3 käyttäen 50×50-hilaa reunat mukaan lukien, siis $48^2 \times 48^2$-matriisia A. Tulosvektori kannattaa muotoilla $48 \times 48-$matriisiksi komennolla `U=reshape(u,48,48)`, jolloin sen voi heti visualisoida komennolla `surf(U)`. Teepä se.

Esitys taululla: Kerro kokeiluistasi ja havainnoista, mitä opit harvoista matriisesita ja miten lopun suorit ja mitä tunnelmia kuva herätti. (Jos joku kohta, kuten kuva ei onnistunut, voit ihmetellä ääneen, miksi.)

5. Määritä sivuiltaan eristetyn sauvan $L = 10$ lämpötila $t = 2$ sekunnin kuluttua alkuhetkestä käyttämällä eksplisiittistä menetelmää numeeriseen approksimointiin. Valitse $h = 1$ ja $k = 0.5$ Alkulämpötila: $f(x) = x - 0.1x^2$ ja sauvan päät pidetään "jäissä". Tehtävä on siis:
$u_t = u_{xx}$ (Olkoon lämpöyhtälön vakio = 1.)
$u(0,t) = 0, u(L,t) = 0$ (Reunaehdot)
$u(x,0) = f(x)$ (Alkuehto).

   Piirrä alkulämpöätilafunktion $f(x)$ ja ratkaisufunktiota $u(x,2)$ approksimoivan lämpötilamurtoviivan kuvaaja.

6. Ratkaise numeerisesti sivuiltaan eristetyn sauvan lämmönjohtumistehtävä, kun sauvan pituus $L = 1$ ja alkulämpötilafunktio $f(x) = x$, kun $0 \le x \le 0.5$, $f(x) = 1 - x$, kun $0.5 < x \le 1$.
   Käytä tällä kertaa implisiittistä *Crank-Nicolsonin* menetelmää valitsemalla askelpituudet $h = 0.2$, $k = 0.04$. Laske ja piirrä ratkaisumurtoviiva ajanhetkellä $t = 0.12$.

Ohjeita, kaavoja

Diff. yhtälöiden numeriikkaa

Annettu diff. yhtälö(systeemi): $y' = \boldsymbol{f}(t, \boldsymbol{y}), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0$

Eulerin menetelmä (EM) $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0$ GKV $= O(h)$
(GKV=**G**lobaali **k**atkaisu**v**irhe)

Klassinen Runge-Kutta (RK4)

$$\begin{cases} k_1 = h\boldsymbol{f}(t_n, \boldsymbol{y}_n) \\ k_2 = h\boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{k_1}{2}) \\ k_3 = h\boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{k_2}{2}) \\ k_4 = h\boldsymbol{f}(t_n + h, \boldsymbol{y}_n + k_3) \\ \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

GKV= $O(h^4)$

Implisiittinen Euler ("Backward Euler", BE) $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})$.

Stabiilisuus *Yhtälöä* sanotaan *stabiiliksi* pisteessä (t, y), jos $f_y(t, y) < 0$, systeemin tapauksessa J_y:n spektri on vasemmassa puolitasossa (om. arvojen reaaliosat < 0), missä J_y on Jacobin matriisi derivoituna y-muuttujien suhteen.

Numeerinen menetelmä on *stabiili*, jos kokonaisvirhe pysyy rajoitettuna. Tämä merkitsee Eulerin menetelmän kohdalla ehtoa:
$-2 < hf_y(t, y) < 0$. Siispä EM voi olla stabiili vain jos yhtälö on stabiili. Siinäkin tapauksessa tulee askelpituusrajoitus: $h < 2/|f_y|$.

Kankeus ("stiffness") tarkoittaa, että ratkaisuilla on eri suuruusluokkaisia aikaskaaloja noudattavia komponentteja. Se ilmenee systeemin kohdalla niin, että J :n ominaisarvot ovat < 0, ja ne ovat keskenään eri suuruusluokkaa. Ongelma tulee siinä, että eksplisiittisellä menetelmällä on aina jokin askelrajoitus. Tällöin nopeasti nollaan menevä komponentti (transientti) määrää stabiilisuuden takia askeleen tolkuttoman pieneksi, vaikka oltaisiin kiinnostuneita vain "henkiin jäävästä", hitaammin muuttuvasta komponentista. "Ei-kankea"ratkaisija voi käyttää järjettömästi aikaa ja pikkuaskelia.

Laplacen yhtälön diskretointimatriisin muodostaminen

Jos suorakulmioalue jaetaan kummasskin suunnassa n:ään osaan (reunat mukaanlukien), on sisäsolmuja $(n-2) \times (n-2)$ kpl. Merk. $m = n - 2$. Laplacen/Poissonin yhtälön diskretoinnissa syntyvä matriisi A on $m^2 \times m^2$-kokoinen. (Tuntemattomia ovat nuo sisäsolmuarvot $u_{i,j}$) Matriisi A voidaan rakentaa $m \times m$ lohkoista B, I, O. B on tridiagonaalimatriisi, jossa on -4 päälävistäjällä ja 1 sivulävistäjillä, I on yksikkömatriisi ja O nollamatriisi. Voit kokeilla MATLAB:lla tyyliin

```
m=3;
keski=4*eye(m,m)
yla=diag(ones(m-1,1),1)   % Ylänauha
ala=diag(ones(m-1,1),-1)  % Alanauha
B=keski+yla+ala
I=eye(m,m); O=zeros(m,m);
A=[B I O;
   I B I;
   O I B]
```

Parempi on opetella käyttämään `sp`-alkuisia funktioita, tässä erityisesti `spdiags`.

```
m=4;
e=ones(m,1);
B=spdiags([e -4*e e],[-1 0 1],m,m)
I=sparse(eye(m,m));
```

Tästä ei ole ihan helppo jatkaa yleisillä parametreilla.

MATLAB:n primitiiveistä lähtöisin elegantein lienee seuraava: e ja I, kuten edellä:

```
B1=spdiags([e -2*e e],[-1 0 1],m,m);
A=kron(B1,I) + kron(I,B1);
```

Omassa `matlab`-hakemistossamme oleva `lapm.m` perustuu toisenlaiseen ajatteluun, rakennetaan koko matriisi pitkistä lävistäjistä `spdiags`:lla. Tämän voisi kuvitella myös olevan tehokas ja muistia säästävä tapa.