Fixpunktsiteration och lösningar av algebraiska ekvationssystem

Johan Jansson

November 22, 2010

Table of contents

Vem är Johan?

Johan Jansson
Pappa till Uno 3 månader
Forskare vid CTL (ctl.csc.kth.se)
`jjan@csc.kth.se`
Doktorerade vid Chalmers 2006
(Tillämpad matematik)
Forskar inom:
Generella metoder och automatiserad mjukvara för FEM
Turbulent fluid-strukturinteraktion
Adaptiv FEM för ODE och PDE
Massivt parallella algoritmer
Mass-fjädersystem

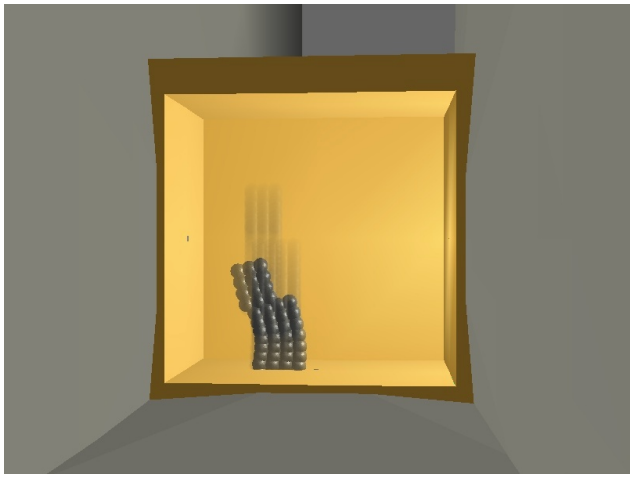
Partikelsystem

- N partiklar (punktmassor) där partikel i har massa m^i, position $x^i = \begin{pmatrix} x_x^i \\ x_y^i \end{pmatrix}$ och hastighet $v^i = \begin{pmatrix} v_x^i \\ v_y^i \end{pmatrix}$ (i 2D).
- Newtons andra lag:

$$\dot{x}^i = v^i$$

$$\dot{v}^i = \frac{F^i}{m^i}.$$

- Parvisa krafter $F^i = \sum_{j=0}^{N} F^{ij}$.



Generell ODE-form

- Skriv på generell ODE-form (vektorform):

$$u = \begin{pmatrix} x_x \\ x_y \\ v_x \\ v_y \end{pmatrix}$$

$$f(u) = \begin{pmatrix} v_x \\ v_y \\ F_x \\ F_y \end{pmatrix}$$

$$\dot{u} = f(u)$$

- Lös med generell ODE-lösare (`solve() från modul 3`, tidsstegning)
- Exempelvis:
  Trapets: $u^{n+1} = u^n + \frac{1}{2}kf(u^n) + \frac{1}{2}kf(u^{n+1})$
  Bakåt Euler: $u^{n+1} = u^n + kf(u^{n+1})$

Partikelsystem i Python

```
# Initial values
x[0, 0] = 0.0  # Particle 0 starts in x=0
...

# Pack values into u
u[0*M:1*M] = x[:, 0]
u[1*M:2*M] = x[:, 1]
u[2*M:3*M] = v[:, 0]
u[3*M:4*M] = v[:, 1]

def f3body(t, u):
  # Unpack values into x and v
  x[:, 0] = u[0*M:1*M]
  x[:, 1] = u[1*M:2*M]
  v[:, 0] = u[2*M:3*M]
  v[:, 1] = u[3*M:4*M]

  a = zeros((M, 2))

  m = 1.0    # Mass
  E = 100.0  # Stiffness coefficient
  B = 4.0    # Damping coefficient
  L = 2.0    # Spring rest length

  for i0 in range(0, M):
      for i1 in range(0, M):
          # Don't compute force with self
          if(i0 == i1):
              continue

          r = norm(x[i1, :] - x[i0, :])
          e = (x[i1, :] - x[i0, :]) / r
          vr = v[i1, :] - v[i0, :]

          # Elastic spring force
          F = E*(r - L)*e

          # Damping spring force
          D = B*dot(vr, e)*e

          # Gravity force
          G = 0 # Add this yourself

          a[i0, :] += (F + D + G) / m

  # Pack values into fval
  fval = zeros(4*M)
  fval[0*M:1*M] = v[:, 0] # (velocities)
  fval[1*M:2*M] = v[:, 1]
  fval[2*M:3*M] = a[:, 0] # (forces/mass)
  fval[3*M:4*M] = a[:, 1]

  return fval
```

Demo 3body

Ekvation i varje tidssteg

- $f(u) = -u$
- Ekvation att lösa i varje tidssteg: $u^{n+1} = u^n + kf(u^{n+1})$
- Notation för ekvation:

$$q = u^{n+1}$$
$$R(q) = 0$$
$$R(q) = -q + u^n + kf(q) = 0$$

- Hur lösa $R(q) = 0$?

Fixpunktsform

- Skriv ekvation $R(q) = 0$ i fixpunktsform $q = g(q)$
  (finns hur många former som helst)
- Fixpunktsiteration: $q_{m+1} = g(q_m)$
- I vårt exempel hade vi redan en fixpunktsform:
  $q_{m+1} = g(q) = u^n + kf(q_m)$
  (notera olika index för tidssteg och iteration)
- Konvergerar iterationen?

Fixpunkt i Python

```
def fixedpoint(g, q0):
    # Choose initial guess, parameters
    q = q0
    TOL = 1.0e-8
    res = 1.0

    # Iterate until convergence
    while(res > TOL):
        s = g(q)
        res = norm(s - q)
        q = s

    return x
```

Fixpunktsform

- Skriv ekvation $R(q) = 0$ i fixpunktsform $q = g(q)$
  (finns hur många former som helst)
- Fixpunktsiteration: $q_{m+1} = g(q_m)$
- I vårt exempel hade vi redan en fixpunktsform:
  $q_{m+1} = g(q) = u^n + kf(q_m)$
  (notera olika index för tidssteg och iteration)
- Konvergerar iterationen?

k=0.1 : snabb konvergens

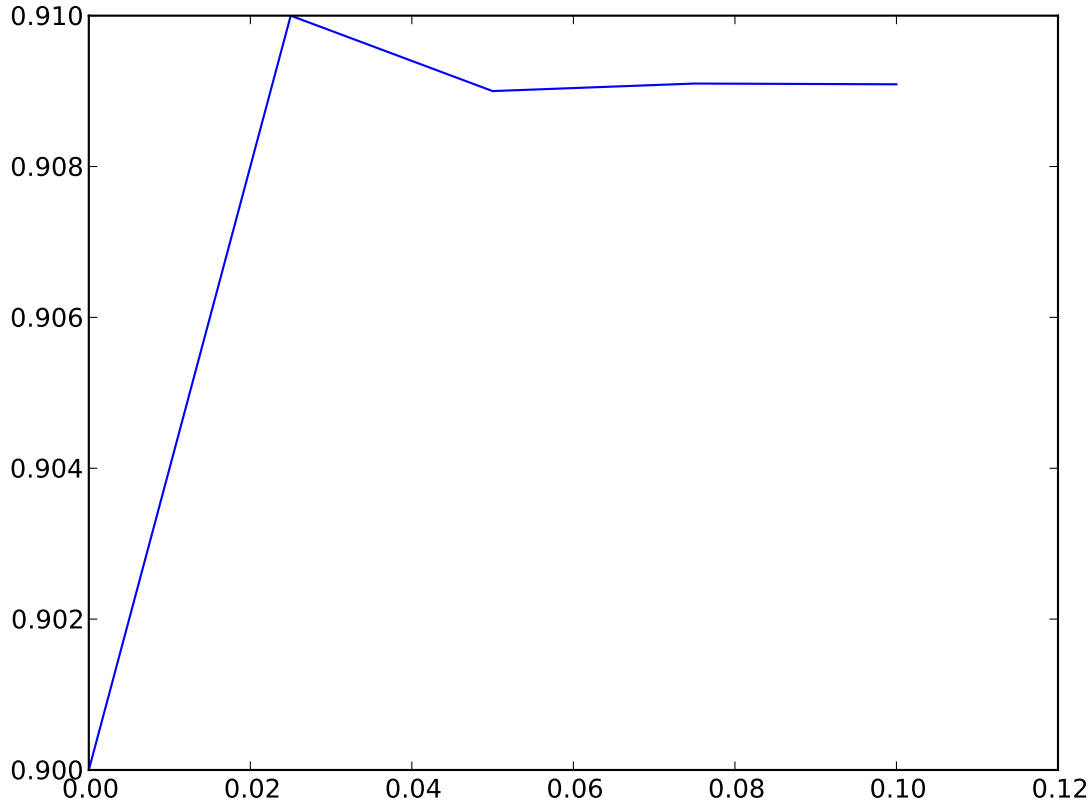
```
e:   0.1
L:   0.1

e:   0.01
L:   0.1

e:   0.001
L:   0.1

e:   0.0001
L:   0.1

e:   1.00000000001e-05
L:   0.100000000001
```



k=2.0 : divergens

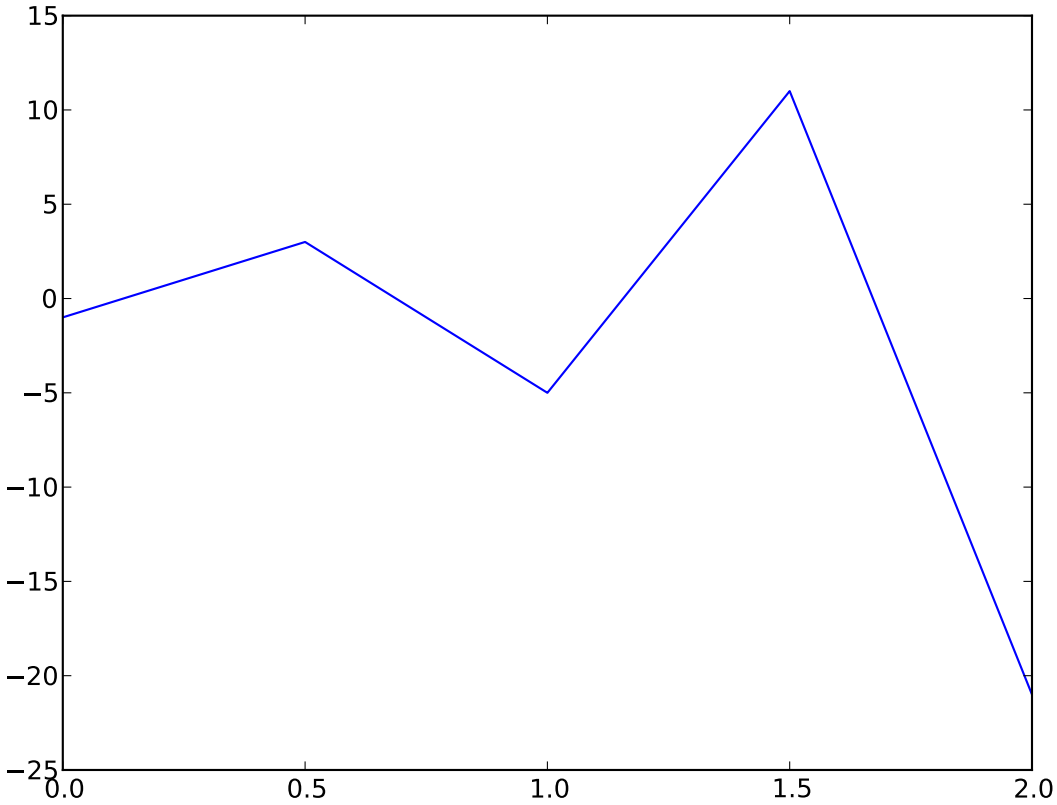
```
e:   2.0
L:   2.0

e:   4.0
L:   2.0

e:   8.0
L:   2.0

e:   16.0
L:   2.0

e:   32.0
L:   2.0
```



k=0.9 : långsam konvergens

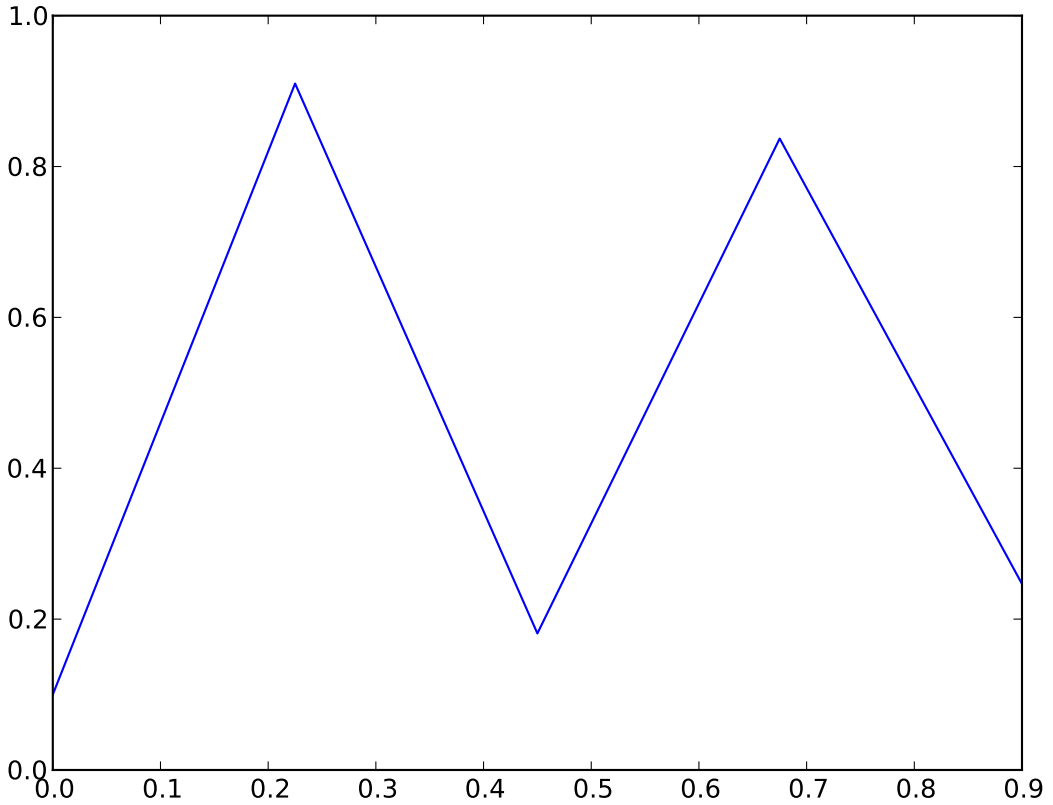
```
e:   0.9
L:   0.9

e:   0.81
L:   0.9

e:   0.729
L:   0.9

e:   0.6561
L:   0.9

e:   0.59049
L:   0.9
```



k=0.9 : långsam konvergens

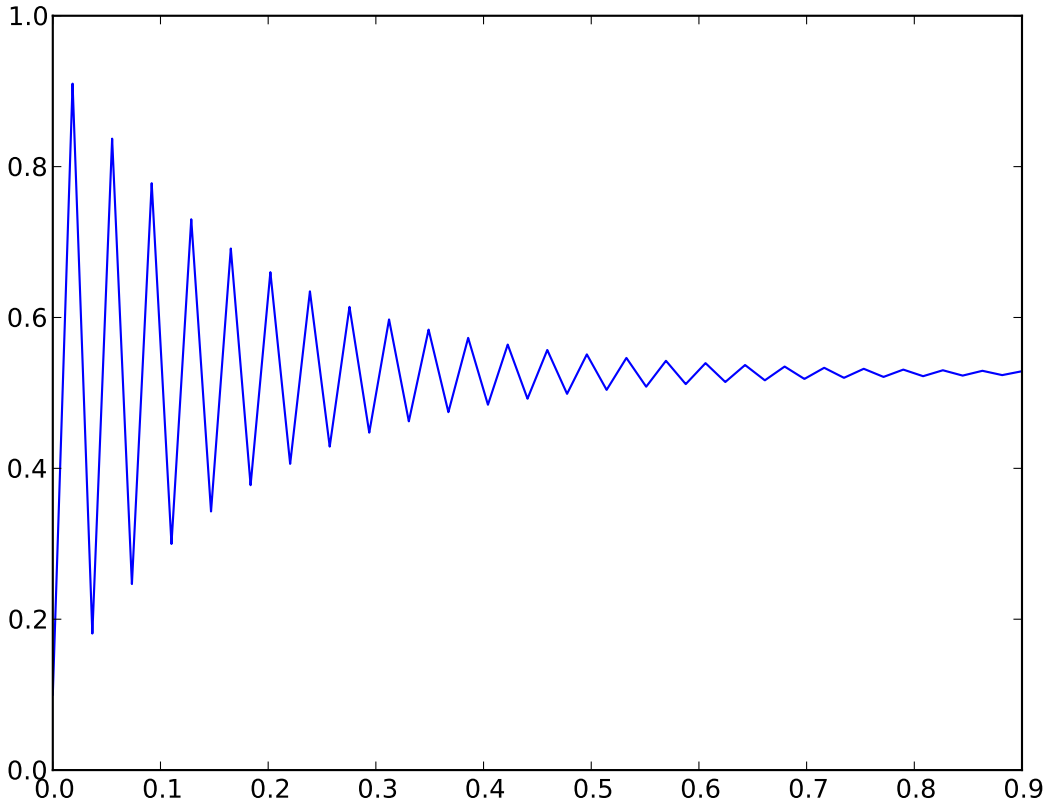
```
e:   0.9
L:   0.9

e:   0.81
L:   0.9

e:   0.729
L:   0.9

e:   0.6561
L:   0.9

e:   0.59049
L:   0.9
...
```



Fixpunkt

- Fixpunktsiteration: $q_{m+1} = g(q_m)$
- $g(q)$ Lipschitzkontinuerlig: $|g(q_{m+1}) - g(q_m)| \leq L|q_{m+1} - q_m|$
-

$$
\begin{aligned}
e^{m+1} &= q^{m+1} - q^m = g(q^m) - g(q^{m-1}) \\
&\Rightarrow \\
|e^{m+1}| &= |g(q^m) - g(q^{m-1})| \leq L|q^m - q^{m-1}| = L|e^m| \\
&\Rightarrow \text{[rekursion]} \\
|e^{m+1}| &\leq L^m|e^1|
\end{aligned}
$$

- $L < 1 \quad \Rightarrow$ fixpunktsiterationen konvergerar med hastighet L
- Vi har att: $|g'(q)| \leq L$ så vi kan använda villkoret $|g'(q)| < 1$.

Tidsstegsvillkor

På tavlan..

Fixpunkt för system

Samma sak, där $g'(q)$ är en matris och $\|g'(q)\|$ en matrisnorm.

Generell algebraisk ekvationslösning

- $R(q) = 0$
- Exempel: $R(q) = x^2 - 2 = 0$ (roten ur två)
- Även system (se linjära system senare t.ex.)

Robustare fixpunkt?

- Vi kan skriva en generell fixpunktsform för ekvation $R(q) = 0$:
  $q = g(q) = q - \alpha R(q)$
- Konvergens: $g'(q) = 1 - \alpha' R(q) - \alpha R'(q)$
- Använd $R(q) = 0$:
  $g'(q) = 1 - \alpha R'(q)$
- Optimal metod: $g'(q) = 0$
  $\Rightarrow$
  $1 - \alpha R'(q) = 0$
  $\Rightarrow$
  $\alpha = \frac{1}{R(q)}$
- Alltså Newtons metod: $q = q - \frac{R(q)}{R'(q)}$

Newton för system

Samma sak, men $R'(q)$ är en matris:

$q_{m+1} = q_m - R'(q_m)^{-1}R(q_m)$

$J = R'(q_m) \Rightarrow$

$Jq_{m+1} = Jq_m - R(q_m)$

Newton i Python

```
def newton(f, x0):
    class LocalData:
        def __init__(self):
            self.f = f

    def g(x, iter, data):
        f = data.f

        # Compute Jacobian
        J = jacobian(f, x)
        # Compute right hand side
        r = dot(J, x) - f(x)
        # Solve linear system
        y = solve(J, r)

        return y

    data = LocalData()

    # Iterate the Newton g(x)
    return fixedpoint(g, x0, data)
```

Newton för tidsstegning

```
def newton_fixedpoint_adapter(g):
    def R(x):
        return x - g(x)

    return R

def g(u):
    t = t0 + k
    return u0 + 0.5 * k * f(t0, u0) + 0.5 * k * f(t, u)

fnewton = newton_fixedpoint_adapter(g)
return newton(fnewton, u0)
```

Demo av Newton för tidsstegning

Fixpunkt för linjära system

Översiktligt.

Jacobi

Jacobi-iteration:

$$Ax = b \Rightarrow [A = D + M] \Rightarrow x = D^{-1}(-Mx + b) = g(x)$$

$$\|g'\| = \|D^{-1}M\| < 1$$

Steepest descent

Steepest Descent:

$$x = x - \alpha(Ax - b) = g(x) \quad (r = Ax - b)$$

$$\|g'\| = \|I - \alpha A\| = 0 \Rightarrow \alpha = \frac{(r, r)}{(r, Ar)}$$

Conjugate Gradient

Conjugate gradient:

Samma som Steepest Descent, men beräkna r som en ortogonalisering mot Krylov-vektorer/rum.

Demo linjära system

Admin

Interaktiv frågestund