NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA)
SCHOOL OF MECHANICAL ENGINEERING
LAB. OF THERMAL TURBOMACHINES
PARALLEL CFD & OPTIMIZATION UNIT (PCOpt/NTUA)

Convergence-Divergence of Iterative Solvers for Linear Systems – Towards the RPM Method

Dr. Kyriakos C. Giannakoglou, Professor NTUA
Dr. Th. Skamagkis

December 2023

The Linear System to be solved

$$A\vec{x} = \vec{b}$$

$$A = \begin{bmatrix} 0.06 & 0.135 & -0.0675 \\ 0.14 & 0.1975 & -0.10375 \\ 0.28 & -0.085 & 0.0325 \end{bmatrix}$$

$$\vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\vec{x}_{sol} = \begin{bmatrix} 11.98 \\ 10.937 \\ 17.7083 \end{bmatrix}$$

1. *Jacobi (standard, with relaxation)*

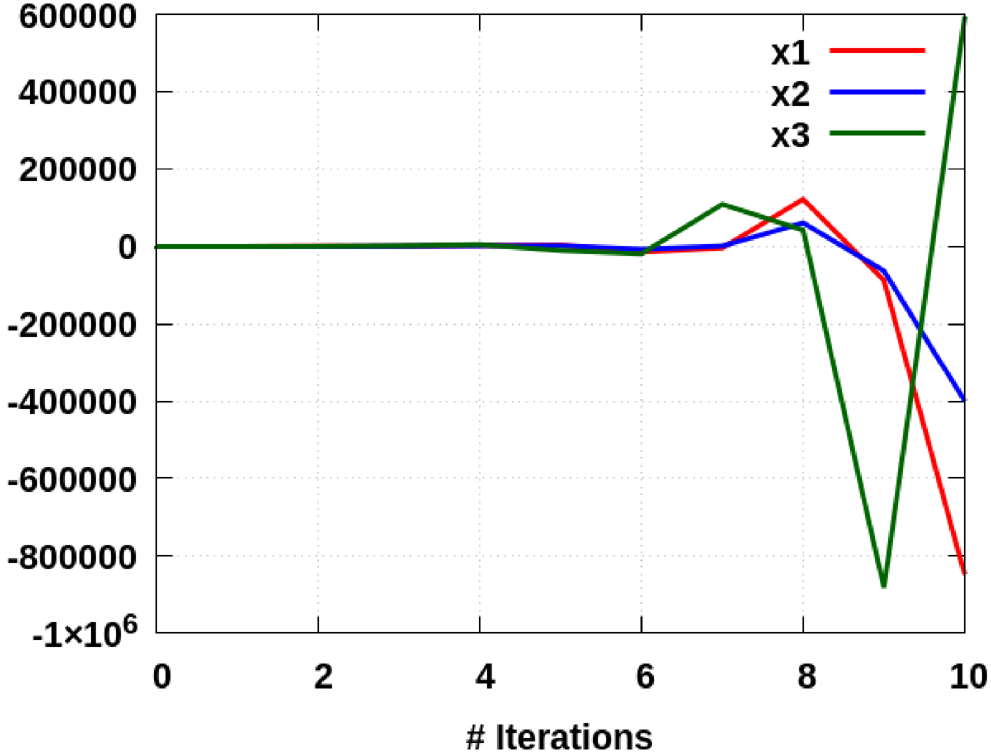
```
do ….   ! iterative loop
    sol1t = ( b(1)-A(1,2)*sol(2)-A(1,3)*sol(3) ) / A(1,1)
    sol2t = ( b(2)-A(2,1)*sol(1)-A(2,3)*sol(3) ) / A(2,2)
    sol3t = ( b(3)-A(3,1)*sol(1)-A(3,2)*sol(2) ) / A(3,3)
    sol(1) = omega*sol1t + (1.d0-omega)*sol(1)
    sol(2) = omega*sol2t + (1.d0-omega)*sol(2)
    sol(3) = omega*sol3t + (1.d0-omega)*sol(3)
enddo
```

1. *Jacobi. ω=1 (iteration counter, x_1, x_2, x_3)*

1	0.1666666667D+02	0.1012658228D+02	0.9230769231D+02
2	0.9772801039D+02	0.4680298604D+02	-0.2479714378D+02
3	-0.1165368387D+03	-0.7217531707D+02	-0.6272488952D+03
4	-0.5265938770D+03	-0.2367691922D+03	0.9075511656D+03
5	0.1570392410D+04	0.8601598796D+03	0.4009873976D+04
6	0.2592415161D+04	0.1003389811D+04	-0.1118757800D+05
7	-0.1482698566D+05	-0.7704553623D+04	-0.1961809573D+05
8	-0.4718445377D+04	0.2146863841D+03	0.1076821208D+06
9	0.1206760083D+06	0.5992203742D+05	0.4130501687D+05
10	-0.8833977356D+05	-0.6383415522D+05	-0.8828587425D+06
11	-0.8495725694D+06	-0.4011494999D+06	0.5942225662D+06
12	0.1571103429D+07	0.9143936758D+06	0.6270326521D+07
13	0.4996748232D+07	0.2180222261D+07	-0.1114407685D+08
14	-0.1744256987D+08	-0.9396155572D+07	-0.3734669578D+08
15	-0.2087366605D+08	-0.7254470405D+07	0.1256999797D+09

…….

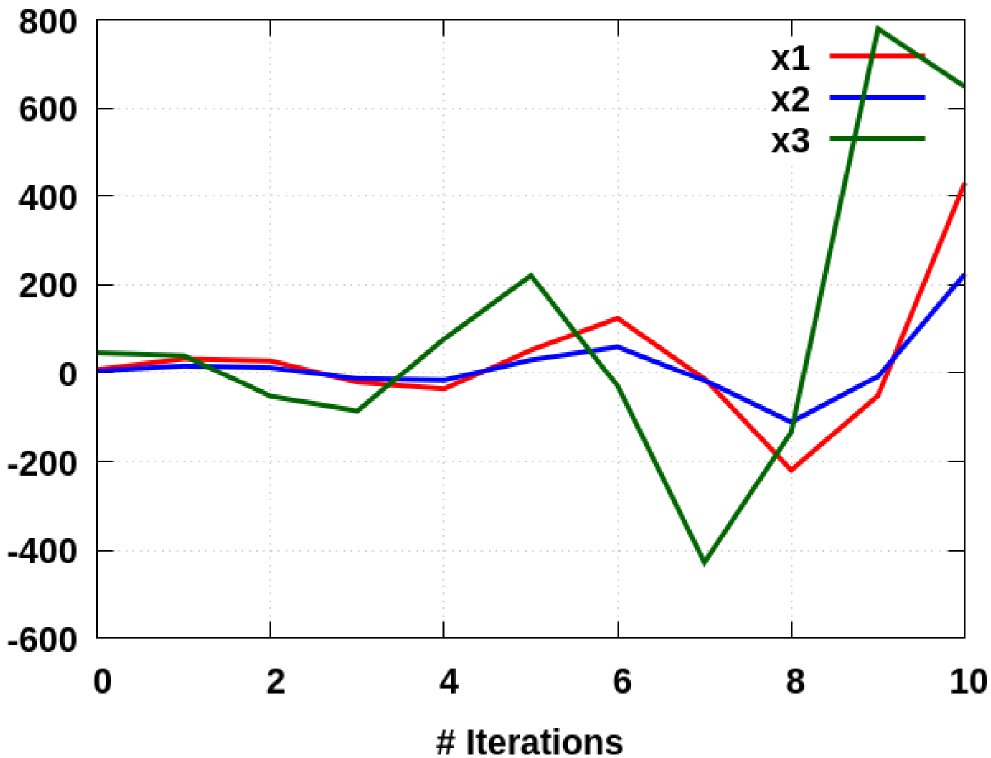
1. *Jacobi. ω=1*


600000
400000
200000
0
-200000
-400000
-600000
-800000
-1×10^6
0
2
4
6
8
10
Iterations
x1
x2
x3

1. *Jacobi. ω=0.5 (iteration counter, x_1, x_2, x_3)*

1	0.8333333333D+01	0.5063291139D+01	0.4615384615D+02
2	0.3276533593D+02	0.1676403765D+02	0.3995456021D+02
3	0.2833089906D+02	0.1232667349D+02	-0.5308965621D+02
4	-0.2123165643D+02	-0.1275913845D+02	-0.8631228180D+02
5	-0.3647912264D+02	-0.1646176501D+02	0.7777212112D+02
6	0.5236007578D+02	0.3018920543D+02	0.2206538192D+03
…			
14	-0.9328621771D+03	-0.4233819537D+03	0.1442550636D+04
15	0.8296416753D+03	0.5029048035D+03	0.4232259064D+04
16	0.2238031991D+04	0.1074103144D+04	-0.7539129414D+03
17	-0.5050927384D+03	-0.4491331021D+03	-0.8566959396D+04
18	-0.4557852956D+04	-0.2290665422D+04	-0.2648869650D+04
19	-0.1183583724D+04	-0.2205752739D+03	0.1536006159D+05
20	0.8304723298D+04	0.4348720239D+04	0.1253625455D+05
…….			

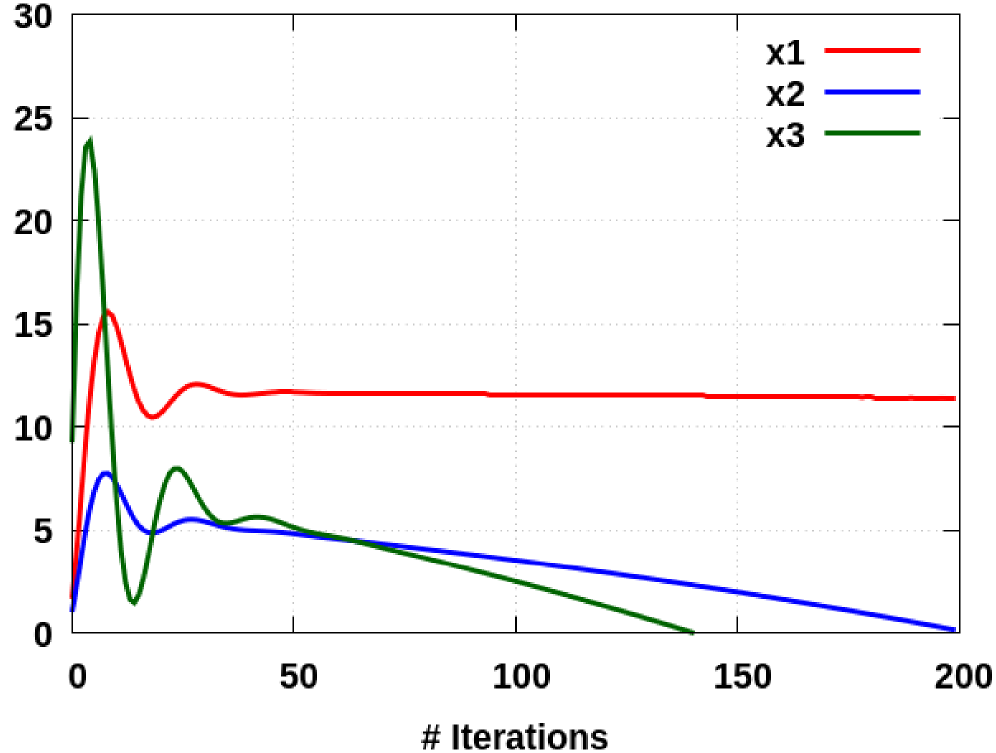
1. *Jacobi. ω=0.5*



1. *Jacobi. ω=0.1 (iteration counter, x_1, x_2, x_3)*

1	0.1666666667D+01	0.1012658228D+01	0.9230769231D+01
2	0.3977280104D+01	0.2290814671D+01	0.1636741318D+02
3	0.6572119442D+01	0.3652264800D+01	0.2113399745D+02
...			
34	0.1174434811D+02	0.5221469316D+01	0.5406453627D+01
35	0.1166997540D+02	0.5163479766D+01	0.5343984946D+01
36	0.1160905989D+02	0.5113279578D+01	0.5336671582D+01
...			
144	0.1148765522D+02	0.2213968316D+01	-0.1922302804D+00
145	0.1148578759D+02	0.2180816739D+01	-0.2602569612D+00
146	0.1148391282D+02	0.2147539155D+01	-0.3285423487D+00
...			
198	0.1137593000D+02	0.2310186469D+00	-0.4261217139D+01
199	0.1137363754D+02	0.1903312237D+00	-0.4344707163D+01
200	0.1137133637D+02	0.1494891778D+00	-0.4428514471D+01
…			

1. *Jacobi. ω=0.1*



1. *Jacobi (standard, with relaxation) – Divergence!!! Why???*

$$A\vec{x} = \vec{b}$$

$$A = D - L - U$$

Jacobi Iteration Matrix:

$$G_J = D^{-1}(L + U)$$

Jacobi with Relaxation ω Iteration Matrix:

$$G_{J,\omega} = (1 - \omega)I + \omega G_J$$

1. *Jacobi (standard, with relaxation) – Divergence!!! Why???*

Jacobi Iteration Matrix:

$$G_J = D^{-1}(L+U)$$

$$G_J = \begin{bmatrix} 0 & -0.225 & 11.25 \\ -0.708860 & 0 & 0.52531645 \\ -8.6153846 & 2.6153846 & 0 \end{bmatrix}$$

$$\overrightarrow{eigenvalues}_J = \begin{bmatrix} 0.10394 + 9.76842i \\ 0.10394 - 9.76842i \\ -0.207879 \end{bmatrix}$$

1. *Jacobi (standard, with relaxation) – Divergence!!! Why???*

Jacobi with Relaxation ω=0.1 Iteration Matrix:

$$G_{J,\omega} = (1-\omega)\mathrm{I} + \omega \mathrm{G}_J$$

$$G_{J,\omega} = \begin{bmatrix} 0.9 & -0.0225 & 1.125 \\ -0.0708860 & 0.9 & 0.052531645 \\ -0.86153846 & 0.26153846 & 0.9 \end{bmatrix}$$

$$\overrightarrow{eigenvalues}_{J,\omega} = \begin{bmatrix} 1.88095 \\ 0.922229 \\ -0.10318 \end{bmatrix}$$

2. *Fixed-Point Iterative Method*

```
do …   ! iterative loop
       sol1t = G(1,1)*sol(1)+G(1,2)*sol(2)+G(1,3)*sol(3) + b(1)
       sol2t = G(2,1)*sol(1)+G(2,2)*sol(2)+G(2,3)*sol(3) + b(2)
       sol3t = G(3,1)*sol(1)+G(3,2)*sol(2)+G(3,3)*sol(3) + b(3)
       sol(1) = omega*sol1t + (1.d0-omega)*sol(1)
       sol(2) = omega*sol2t + (1.d0-omega)*sol(2)
       sol(3) = omega*sol3t + (1.d0-omega)*sol(3)
enddo  !  iter
```

$$A\vec{x} = \vec{b} \Rightarrow \vec{x} = G\vec{x} + \vec{b}$$

$$G = I - A \quad \text{or} \quad A = I - G$$

2. *Fixed-Point Iterative Method*

$$\vec{x}^{n+1} = G\vec{x}^{n} + \vec{b}$$

$$G = I - A$$

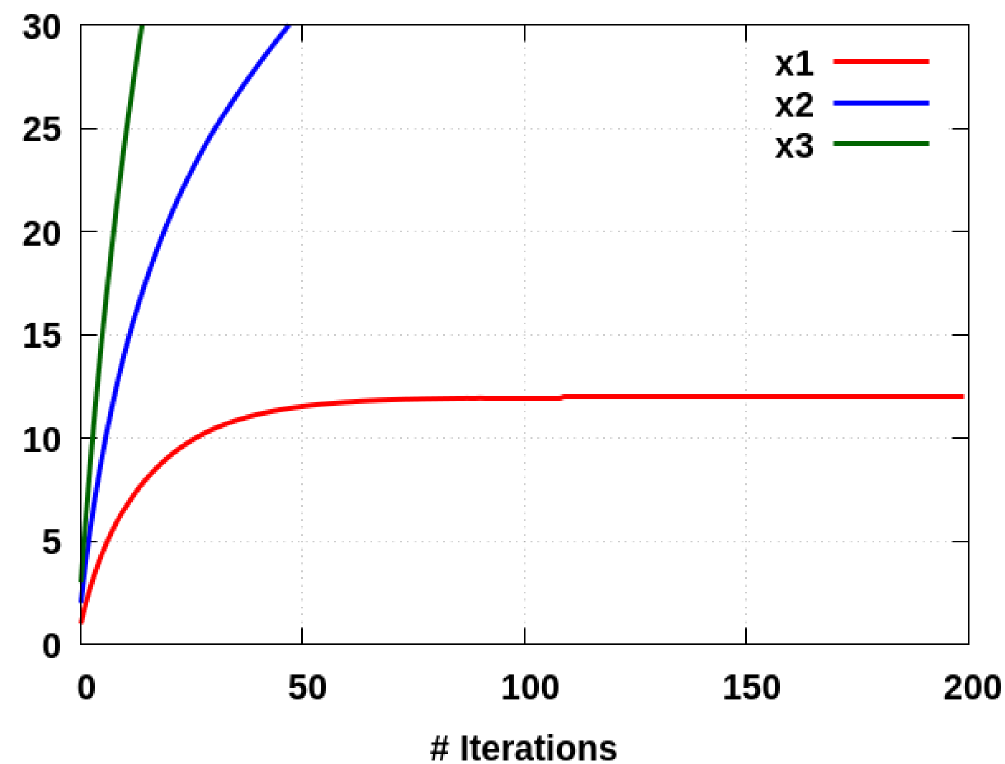
$$G = \begin{bmatrix} 0.94 & -0.135 & 0.0675 \\ -0.14 & 0.8025 & 0.10375 \\ -0.28 & 0.085 & 0.9675 \end{bmatrix}$$

Plus relaxation, if necessary……

$$\vec{x}^{*} = G\vec{x}^{n} + \vec{b}$$

$$\vec{x}^{n+1} = \omega\vec{x}^{*} + (1\text{-}\omega)\,\vec{x}^{n}$$

2. *Fixed-Point Iterative Method. ω=1.0*



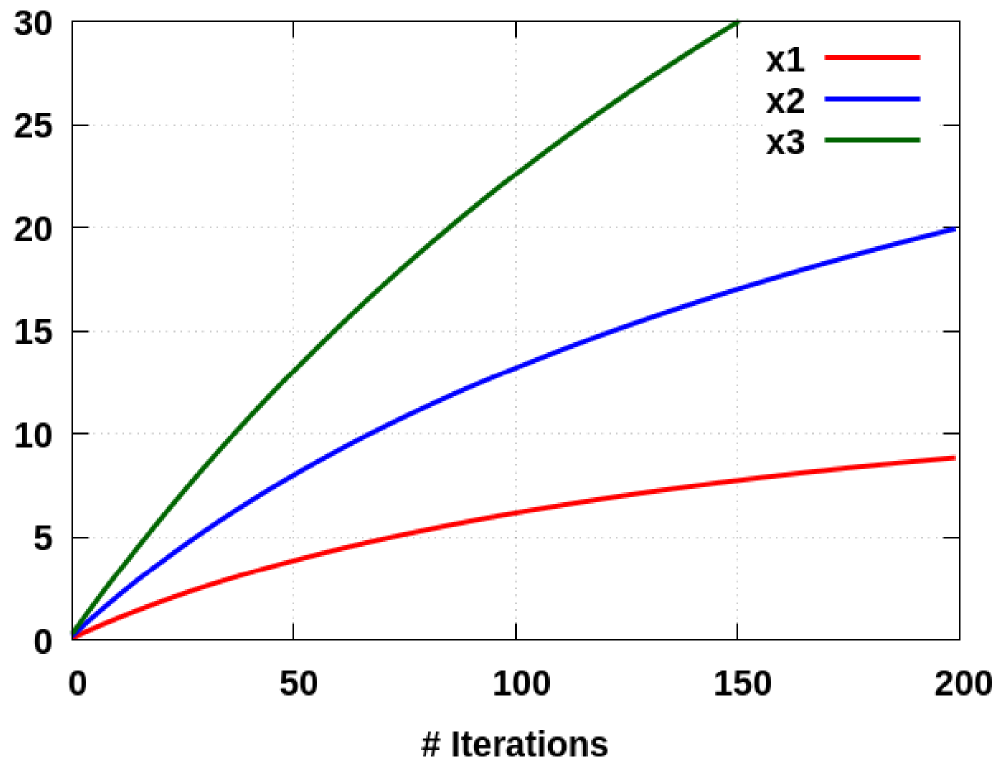
2. *Fixed-Point Iterative Method. ω=1.0 (iteration counter, x_1, x_2, x_3)*

...

195	0.1197910673D+02	0.9794912958D+02	0.1917315925D+03
196	0.1197911033D+02	0.9881925427D+02	0.1934718419D+03
197	0.1197911371D+02	0.9969807970D+02	0.1952294927D+03
198	0.1197911689D+02	0.1005856929D+03	0.1970047192D+03
199	0.1197911987D+02	0.1014821818D+03	0.1987976970D+03
200	0.1197912268D+02	0.1023876352D+03	0.2006086037D+03

…

2. *Fixed-Point Iterative Method. ω=0.1*



2. *Fixed-Point Iterative Method. ω=0.1 (iteration counter, x_1, x_2, x_3)*

...

195	0.8743856900D+01	0.1966135241D+02	0.3519255583D+02
196	0.8763515253D+01	0.1971574947D+02	0.3530047352D+02
197	0.8783049740D+01	0.1976991661D+02	0.3540795243D+02
198	0.8802461246D+01	0.1982385557D+02	0.3551499548D+02
199	0.8821750648D+01	0.1987756804D+02	0.3562160560D+02
200	0.8840918813D+01	0.1993105572D+02	0.3572778570D+02

…

2. *Fixed-Point Iterative Method– Divergence!!! Why???*

Jacobi with Relaxation ω=0.1 Iteration Matrix:

$$G=\begin{bmatrix} 0.94 & -0.135 & 0.0675 \\ -0.14 & 0.8025 & 0.10375 \\ -0.28 & 0.085 & 0.9675 \end{bmatrix}$$

$$\overrightarrow{eigenvalues} = \begin{bmatrix} 1.01 \\ 0.94 \\ 0.76 \end{bmatrix}$$

3. *A Decoupled Fixed Point Iteration Solver*

Eigenvector matrix of G (of the FPI algorithm):
(each column of V stands for an eigenvector)

$$V = \begin{bmatrix} 0. & 0.25 & 1.5 \\ 0.5 & 0.5 & 2.5 \\ 1. & 1. & 1. \end{bmatrix}$$

$$G = V \Lambda V^{-1}$$

Its inverse:

$$V^{-1} = \begin{bmatrix} -4. & 2.5 & -0.25 \\ 4. & -3. & 1.5 \\ 0 & 0.5 & -0.25 \end{bmatrix}$$

Also, with:

$$\Lambda = \begin{bmatrix} 1.01 & 0 & 0 \\ 0 & 0.94 & 0 \\ 0 & 0 & 0.76 \end{bmatrix}$$

3. *A Decoupled Fixed Point Iteration Solver*

$$\vec{x} = G\vec{x} + \vec{b} \rightarrow \vec{x} = V\Lambda V^{-1}\vec{x} + \vec{b} \rightarrow$$
$$V^{-1}\vec{x} = \Lambda V^{-1}\vec{x} + V^{-1}\vec{b} \rightarrow \vec{z} = \Lambda\vec{z} + \vec{\beta}$$

where:

$$\vec{z} = V^{-1}\vec{x} \quad \& \quad \vec{\beta} = V^{-1}\vec{b}$$

After computing the new unknown (z), we readily return to x, as follows:

$$\vec{x} = V\vec{z}$$

3. *A Decoupled Fixed Point Iteration Solver*

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1.01 & 0 & 0 \\ 0 & 0.94 & 0 \\ 0 & 0 & 0.76 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 0.25 \\ 2.5 \\ 0.25 \end{bmatrix}$$

```
call matrvec(vminv,b,bnew)
call matrvec(vminv,sol,res)

do …   ! iterative loop
       res(1:3)=ei(1:3)*res(1:3)+bnew(1:3)
       call matrvec(vm,res,sol)   → only for printout…
enddo  !  Iter
```

$$\vec{z} = \Lambda \vec{z} + \vec{\beta}$$

$$\vec{x} = V \vec{z}$$

3. *A Decoupled Fixed Point Iteration Solver*

$$\vec{x} = V\,\vec{z} \qquad \vec{z} = V^{-1}\,\vec{x}$$

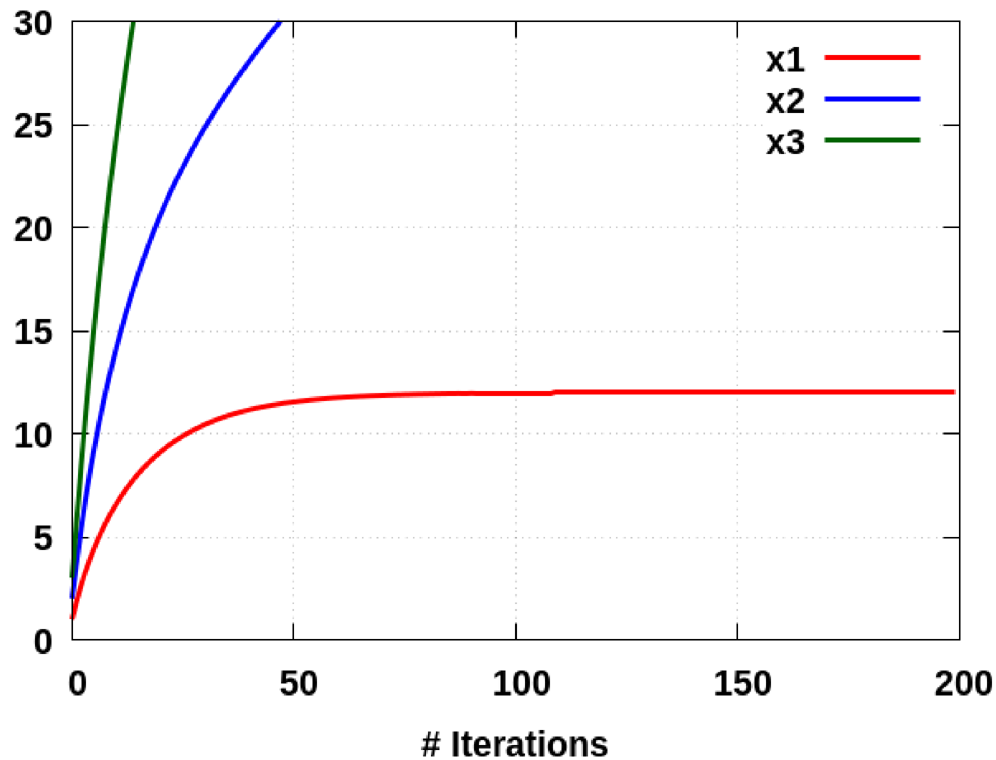
$\vec{x}$

...			
195	0.1197910673D+02	0.9794912958D+02	0.1917315925D+03
196	0.1197911033D+02	0.9881925427D+02	0.1934718419D+03
197	0.1197911371D+02	0.9969807970D+02	0.1952294927D+03
198	0.1197911689D+02	0.1005856929D+03	0.1970047192D+03
199	0.1197911987D+02	0.1014821818D+03	0.1987976970D+03
200	0.1197912268D+02	0.1023876352D+03	0.2006086037D+03

$\vec{z}$

...			
195	0.1490234989D+03	0.4166642694D+02	0.1041666667D+01
196	0.1507637339D+03	0.4166644132D+02	0.1041666667D+01
197	0.1525213712D+03	0.4166645484D+02	0.1041666667D+01
198	0.1542965849D+03	0.4166646755D+02	0.1041666667D+01
199	0.1560895508D+03	0.4166647950D+02	0.1041666667D+01
200	0.1579004463D+03	0.4166649073D+02	0.1041666667D+01

3. *A Decoupled Fixed Point Iteration Solver*



4. *Decoupled Newton-FPI-FPI*

```
call matrvec(vminv,b,bnew)
call matrvec(vminv,sol,res)

   do …..   ! iterative loop
     res(1)=res(1)-(res(1)-ei(1)*res(1)-bnew(1))/(1.d0-ei(1))
     res(2:3)=ei(2:3)*res(2:3)+bnew(2:3)
     call matrvec(vm,res,sol)    → only for printout…
   enddo  !  iter
```

Only for i=1

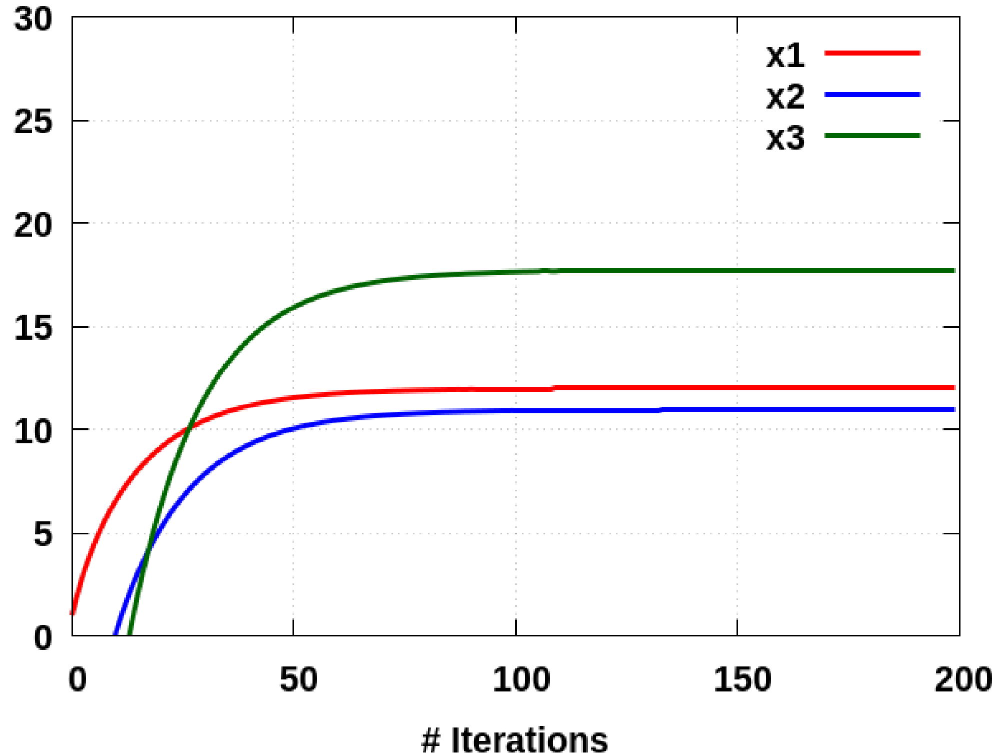
$$\vec{z} = \Lambda\vec{z} + \vec{\beta}$$

$$F(z_i) = z_i(1-\lambda_i) - \beta_i = 0$$

$$z_i^{n+1} = z_i^n - \frac{F(z_i^n)}{F^{/}(z_i^n)}$$

$$\frac{dF(z_i)}{dz_i} = F^{/} = (1-\lambda_i)$$

4. *Decoupled Newton-FPI-FPI*



5. *Decoupled All – Newton-Raphson*

```
call matrvec(vminv,b,bnew)
call matrvec(vminv,sol,res)

 do  ….  ! iterative loop
   res(1:3)=res(1:3)-(res(1:3)-ei(1:3)*res(1:3)-bnew(1:3))  / (1.d0-ei(1:3))
   call matrvec(vm,res,sol)    →  only for printout…
enddo  !  iter
```

For i=1,2,3

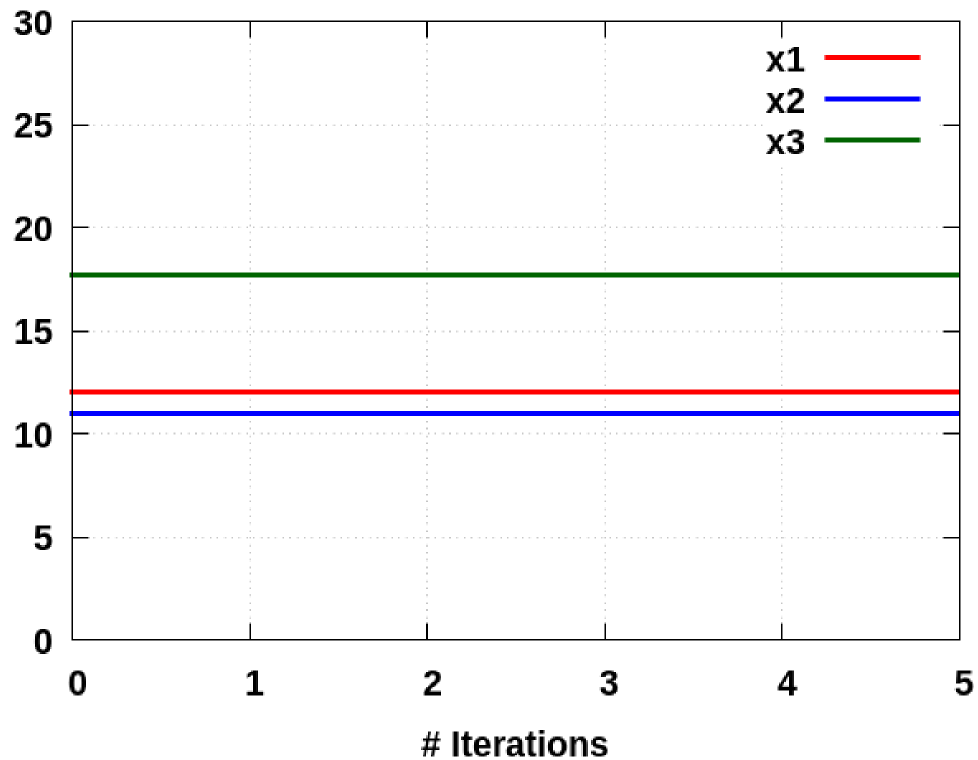
$$\vec{z} = \Lambda\vec{z} + \vec{\beta}$$

$$F(z_i) = z_i(1-\lambda_i) - \beta_i = 0$$

$$z_i^{n+1} = z_i^n - \frac{F(z_i^n)}{F^{/}(z_i^n)}$$

$$\frac{dF(z_i)}{dz_i} = F^{/} = (1-\lambda_i)$$

5. *Decoupled All – Newton-Raphson*



6. *Recursive Projection Method, RPM* **(part 1)**

```
omega=1.0d0
   maxit1=100
   do iter=1,maxit1    ! iterative loop / first phase (Jacobi)
     res(1) = G(1,1)*sol(1)+G(1,2)*sol(2)+G(1,3)*sol(3) + b(1)
     res(2) = G(2,1)*sol(1)+G(2,2)*sol(2)+G(2,3)*sol(3) + b(2)
     res(3) = G(3,1)*sol(1)+G(3,2)*sol(2)+G(3,3)*sol(3) + b(3)
     if(iter.gt.1) ds(1:3)=res(1:3)-sol(1:3)
     sol(1:3)=res(1:3)
   enddo  !  Iter

     dsnorm=dsqrt(ds(1)**2+ds(2)**2+ds(3)**2)
     ds(1:3) = ds(1:3)/dsnorm   !  =  Z - chosen basis
     call matrvec(G,ds,az)
     h =ds(1)*az(1)+ds(2)*az(2)+ds(3)*az(3)   !  corresp. eigenvalue
```

6. *Recursive Projection Method, RPM* (part 2)

```
do iter=maxit1+1,maxiter    ! iterative loop / 2nd phase (RPM)
   res(1) = G(1,1)*sol(1)+G(1,2)*sol(2)+G(1,3)*sol(3) + b(1)
   res(2) = G(2,1)*sol(1)+G(2,2)*sol(2)+G(2,3)*sol(3) + b(2)
   res(3) = G(3,1)*sol(1)+G(3,2)*sol(2)+G(3,3)*sol(3) + b(3)
   zetatemp = ds(1)*res(1)+ds(2)*res(2)+ds(3)*res(3)
   zetaold    = ds(1)*sol(1)+ds(2)*sol(2)+ds(3)*sol(3)
   p(1:3) = ds(1:3)*zetatemp
   q(1:3) = res(1:3) - p(1:3)
   zetanew=zetaold-(zetaold-zetatemp)/(1.d0-h)
   p(1:3) = ds(1:3)*zetanew
   sol(1:3) = p(1:3) + q(1:3)
enddo  !  iter
```

6. *Recursive Projection Method, RPM*