Gleichungslöser I

Den Gleichungslösern oder Solvern, wie sie im Englischen genannt werden, kommt bei der Finite Elemente Analyse eine ganz besondere Bedeutung zu. Da der Aufwand zur Lösung der FEA-Gleichungssysteme, wie man aufgrund theoretischer Betrachtungen zeigen kann, nicht nur quadratisch (wie man meinen sollte), sondern sogar zur 3. Potenz der Anzahl der Freiheitsgrade steigt, kommt man mit den üblichen Verfahren nicht weit. FEA-Solver müssen auf höchste Effizienz getrimmt sein, dabei aber gleichzeitig hohe numerische Stabilität aufweisen. Tatsächlich steckt viel Know-How in den Solvern kommerzieller Großsysteme, das dann den entscheidenden Vorsprung bringt.

Gleichungslöser II

Grundsätzlich unterscheidet man

1. direkte Verfahren
2. iterative Verfahren

Diese beiden grundlegend verschiedenen Verfahren werden wir jetzt betrachten, wobei zuvor ein sehr interessanter Nebeneffekt nochmals erwähnt werden soll: Gleichungslöser *per se* sind bei der Finite Elemente Analyse nur die halbe Wahrheit; erst ihre Kombination mit den passenden Speicherverfahren macht sie zu leistungsfähigen Werkzeugen:

Gleichungslöser III

	Speicher-anforderung	Empfindlichkeit auf Knotennumerierung	Gleichungs-löser	Programmier-aufwand
Band-speicherung	meist groß	hoch	alle	minimal
Skyline-Speicherung	besser als Band	mittel	alle	mittel
Jennings-Speicherung	Wie Skyline	mittel	alle	höher als Skyline
Nicht-Null-elemente	minimal	fast unempfindlich	Nur Iterations-Solver	sehr hoch

Direkte Verfahren I

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$$

$$\sum_j a_{ij} x_j = b_i$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{ns} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Bringe durch elementare Umformungen wie

1. Multiplikationen einer Matrixzeile mit einem Faktor
2. Addition oder Subtraktion von Vielfachen einer Matrixzeile
3. Vertauschen von Zeilen

auf Dreiecksgestalt:

$$\begin{pmatrix} a'_{11} & a'_{12} \cdots & a'_{1n} \\ & a'_{22} \cdots & a'_{2n} \\ & \ddots & \vdots \\ & & a'_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{pmatrix}$$

Direkte Verfahren II

Rückrechnen:

Aus Zeile n:

$$x_n = \frac{b'_n}{a'_{nn}}$$

Aus Zeile n-1:

$$a'_{n-1,n-1} \cdot x_{n-1} + a'_{n-1,n} \cdot x_n = b'_{n-1}$$

$$x_{n-1} = \frac{b'_{n-1} - x_n \cdot a'_{n-1,n}}{a'_{n-1,n-1}}$$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

zu aufwendig, bringt's nicht

Direkte Verfahren III

Bei den Solvern vom Gauß-Typ wird zunächst eine Dreieckszerlegung vorgenommen:

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{R} \quad \Rightarrow \quad a_{ij} = \sum_k \ell_{ik} r_{kj}$$

$$\ell_{ik} = 0 \quad \text{für } i < k$$

$$r_{kj} = 0 \quad \text{für } k > j$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{ab} \\ a_{21} & a_{22} & & \vdots \\ \vdots & \vdots & & \vdots \\ a_{na} & a_{n2} & & a_{nn} \end{pmatrix} = \begin{pmatrix} \ell_{11} & & 0 & \\ \ell_{21} & \ell_{22} & & \\ \vdots & & & \\ \ell_{n1} & \ell_{n2} & & \ell_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & \vdots \\ \vdots & & & \vdots \\ 0 & & & r_{nn} \end{pmatrix}$$

Direkte Verfahren IV

Diese Zerlegung nennt man daher auch *LU-Zerlegung* (Lower-Upper-decomposition) im Englischen bzw. *LR* (Links-Rechts-Zerlegung) im Deutschen.

Das Lösen des Gleichungssystems $\boldsymbol{Ax} = \boldsymbol{b}$ erfolgt in drei Schritten:

1. Schritt: LR- Zerlegung der Matrix $\boldsymbol{A}$, d.h. $\boldsymbol{Ax} = \boldsymbol{LRx} = \boldsymbol{b}$

2. Schritt: Lösen des Gleichungssystems $\boldsymbol{Ly} = \boldsymbol{b}$ mit $\boldsymbol{y}$ Hilfsvektor

3. Schritt: Lösen des Gleichungssystems $\boldsymbol{Rx} = \boldsymbol{y}$

Dabei ist es in der FEA- Rechenpraxis so, daß vielleicht 90% des Aufwands auf Schritt 1 entfallen und nur jeweils 5% auf die Schritte 2 und 3.

Direkte Verfahren V

$$\ell_{ij} = \frac{1}{r_{jj}}\left(a_{ij} - \sum_k \ell_{ik} r_{kj}\right)$$ für $k < j < i$

$$\ell_{ii} r_{ii} = a_{ii} - \sum_k \ell_{ik} r_{ki}$$ für $k < i$

$$r_{ij} = \frac{1}{\ell_{ii}}\left(a_{ij} - \sum_k \ell_{ik} r_{kj}\right)$$ für $k < i < j$

$r_{ii} = 1$:„Crout- Zerlegung“

$l_{ii} = 1$:„Doolittle- Zerlegung“

$l_{ii} = r_{ii}$:„Cholesky- Zerlegung“

Cholesky-Verfahren I

Die dritte Variante, die sog. Cholesky- Zerlegung (nach *André-Louis Cholesky*, 1875-1918) ist in der Praxis sehr beliebt, und viele FEA-Programme arbeiten damit - auch Z88.

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T \quad \Rightarrow \quad \ell_{ij} = r_{ji}$$

$$r_{ii} = \sqrt{a_{ii} - \sum_k r_{ki}^2}\ , \quad \text{für} \quad k < i$$

$$r_{ij} = \frac{1}{r_{ii}}\left(a_{ij} - \sum_k r_{ki} r_{kj}\right), \quad \text{für} \quad k < i < j$$

Cholesky-Verfahren II

und jetzt auflösen:

$$y_i = \frac{1}{r_{ii}}\left(b_i - \sum_k r_{ki} y_k\right), \quad \text{für} \quad k < i \quad \text{und} \quad i,k = 1,2,\ldots,n$$

$$x_i = \frac{1}{r_{ii}}\left(y_i - \sum_k r_{ik} x_k\right), \quad \text{für} \quad k > i \quad \text{und} \quad i,k = n,n-1,\ldots,1$$

Das Cholesky-Verfahren hat den Vorteil, daß es numerisch sehr stabil ist, weil durch das Radizieren eine stabilisierende Wirkung eintritt: Große Zahlen werden kleiner und kleine Zahlen (zwischen 0 und 1) werden größer. Das ist ein sehr erwünschter Effekt, weil die Gleichungssysteme bei der Finite Elemente Analyse nicht selten schlecht konditioniert sind.

Cholesky-Verfahren III

Auf den ersten Blick sieht es so aus, als ob man Speicher für ***A*** und ***R*** vorhalten müsse. Aber tatsächlich kann man ***A*** Zug um Zug durch ***R*** überspeichern, so daß kein Zusatzspeicher nötig ist. Das folgende, dahingehend modifizierte Cholesky- Verfahren mit *in-situ Speicherung* ist *Argyris* entnommen:

$$r_{11} = \sqrt{a_{11}}$$

$$r_{ij} = \frac{a_{ij}}{r_{ii}}$$

$$a_{kj} = a_{kj} - r_{ik} \cdot r_{ij} \quad , k = i+1, j$$

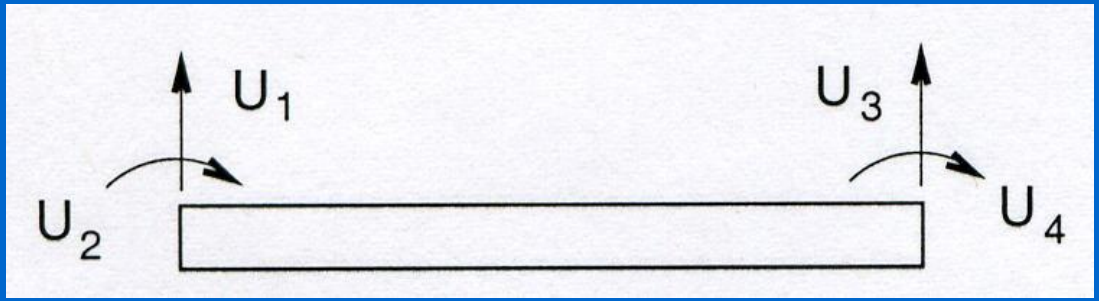
$$r_{jj} = \sqrt{a_{jj}}$$

choy88.c

i=1,j-1

j=2,n

Konditionierung, Skalierung I



Oha

z.B. l= 1.000

$$EI \cdot \begin{bmatrix} \frac{12}{\ell^3} & \frac{-6}{\ell^2} & \frac{-12}{\ell^3} & \frac{-6}{\ell^2} \\ \frac{-6}{\ell^2} & \frac{4}{\ell} & \frac{6}{\ell^2} & \frac{2}{\ell} \\ \frac{-12}{\ell^3} & \frac{6}{\ell^2} & \frac{12}{\ell^3} & \frac{6}{\ell^2} \\ \frac{-6}{\ell^2} & \frac{2}{\ell} & \frac{6}{\ell^2} & \frac{4}{\ell} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

$$a_{11} = \frac{12}{10^9} = a_{33}$$

$$a_{22} = \frac{4}{10^3} = a_{44}$$

Konditionierung, Skalierung II

A skalieren:

$$d_i = \frac{1}{\sqrt{a_{ii}}}$$

skalierte Matrix:

$$\hat{\boldsymbol{A}} = \boldsymbol{D}\,\boldsymbol{A}\,\boldsymbol{D}$$

$$\hat{a}_{ij} = d_i a_{ij} d_j$$

skalierte Rechte Seite:

$$\hat{\boldsymbol{b}} = \boldsymbol{D}\,\boldsymbol{b}$$

$$\hat{b}_i = d_i b_i$$

Skalierung rückgängig machen:

$$\boldsymbol{x} = \boldsymbol{D}\,\hat{\boldsymbol{x}}$$

$$x_i = d_i \hat{x}_i$$

choy88.c

Iterationsverfahren I

Der Hauptvorteil der iterativen Verfahren liegt beim Einsatz für die FEA darin, daß

1. sie fast unempfindlich gegen ungünstige Knotennumerierungen sind
2. und mit einem Minimum an Speicher auskommen, wenn sie mit dem Verfahren der Speicherung der Nicht-Nullelemente kombiniert werden.

In der Literatur wird oft behauptet, daß Iterationssolver bei großen Gleichungssystemen schneller arbeiten als Direkte Solver. Dies ist nach meiner Beobachtung aufgrund vieler numerischer Experimente aber nur teilweise richtig, und zwar können die Iterationssolver durchaus schneller sein; dies liegt aber nicht in ihrer mathematischen Struktur begründet, sondern – besonders in der Kombination mit dem Verfahren der Speicherung der Nicht-Nullelemente – daß durch die durchweg dünne Besetzung weniger mathematische Operationen wie Additionen und Multiplikationen ausgeführt werden müssen. Und zwar werden Blindoperationen, also z.B. Addition von Null, dadurch vermieden. Bei dicht besetzten Gleichungssystemen wird man sogar eher finden, daß die Direkten Solver schneller sind.

Iterationsverfahren II

Einen Nachteil haben Iterationssolver aber definitiv: Man weiß nie, wann sie fertig sind bzw. eine "genaue" Lösung erreicht haben. Darüber werden wir noch sprechen müssen.

Wir gehen wieder von einem Gleichungssystem aus:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{ns} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$$

$$\sum_j a_{ij} x_j = b_i$$

Iterationsverfahren III

1. Zerlege ***A*** wie folgt: $\boldsymbol{A} = \boldsymbol{N} - \boldsymbol{P} \quad \text{mit } \det \boldsymbol{N} \neq 0$

2. Definiere die sog. Iterationsmatrix ***M*** und den sog. Konstantenvektor ***d*** :

$$\boldsymbol{M} = \boldsymbol{N}^{-1}\boldsymbol{P} \quad \text{und} \quad \boldsymbol{d} = \boldsymbol{N}^{-1}\boldsymbol{b}$$

3. Bilde die Vektorfolge, beginne mit einem Startvektor $\boldsymbol{x}^{(0)}$:

$$\boldsymbol{x}^{(\nu)} = \boldsymbol{M}\,\boldsymbol{x}^{(\nu-1)} + \boldsymbol{d} \quad \text{mit} \quad \nu = 1,2,\ldots$$

4. Wenn dieses System konvergiert, dann ist dies die Lösung:

$$\boldsymbol{x} = \boldsymbol{x}^{(\nu)}\ ,\ \nu \to \infty$$

Iterationsverfahren IV

Umgekehrt kann man zeigen, daß das Iterationsverfahren dann konvergiert, wenn der Spektralradius (das ist der betragsgrößte Eigenwert einer Matrix)

$$\rho(\boldsymbol{M}) < 1$$

"Ein mathematisches Problem heißt gut konditioniert, wenn eine geringe relative Störung der Eingabedaten des Problems auch nur eine geringe relative Änderung des Ergebnisses (d.h. der Lösung des Problems) zur Folge hat. Hat eine geringe relative Änderung der Eingabedaten eine große relative Änderung des Ergebnisses zur Folge, dann nennt man das Problem schlecht konditioniert."

"Gegeben sei ein mathematisches Problem und ein numerisches Verfahren zur Lösung dieses Problems. Das numerische Verfahren heißt gut konditioniert, wenn seine Lösung die exakte Lösung des mathematischen Problems ist, das aus dem ursprünglichen mathematischen Problem durch geringe relative Änderungen der Eingabedaten hervorgeht. Andernfalls heißt das numerische Verfahren schlecht konditioniert."

Iterationsverfahren V

Vektornormen:

Maximumnorm: $$\|\boldsymbol{x}\|_\infty = \max |x_k| \text{, mit } k = 1,..,n$$

L_1-Norm: $$\|\boldsymbol{x}\|_1 = \sum_{k=1}^{n} |x_k|$$

Euklidische Norm: $$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{k=1}^{n} x_k^2}$$

Iterationsverfahren VI

Matrixnormen:

Spalten- oder „1“ - Norm:

$$\|\boldsymbol{A}\|_1 = \max \sum_{i=1}^{n} |a_{ik}| \text{, mit } k = 1,..,n$$

Zeilen- oder „∞- Norm:

$$\|\boldsymbol{A}\|_\infty = \max \sum_{k=1}^{n} |a_{ik}| \text{, mit } i = 1,..,n$$

Spektral- oder „2“- Norm:

$$\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^*\mathbf{A})}$$

$$\mathbf{A}^* = \mathbf{A}^T$$

für Matrizen mit reellen Elementen:

Iterationsverfahren VII

Spezielle Matrizen: Diagonalmatrix:

$$\boldsymbol{D} = \begin{pmatrix} a_{1,1} & 0 & 0 & 0 \\ 0 & a_{2,2} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{n,n} \end{pmatrix} = \boldsymbol{D} = a_{ij}\,\delta_{ij} \quad \text{mit } \delta_{ij} : \text{Kronecker} - \text{Symbol}$$

Strikt untere Links-Dreiecksmatrix ***E*** , strikt obere Rechts-Dreiecksmatrix ***F***:

$$-\boldsymbol{E} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 \\ \dots & \dots & \dots & 0 \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{pmatrix} \quad -\boldsymbol{F} = \begin{pmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ 0 & 0 & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Iterationsverfahren VIII

Das Jacobi-Verfahren, J-Verfahren, Gesamtschritt-Verfahren, ***N*** wie folgt gewählt:

$$\boldsymbol{N} = \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{nn} \end{pmatrix} = \boldsymbol{D} \quad \Rightarrow \quad \boldsymbol{M} = \boldsymbol{I} - \boldsymbol{N}^{-1}\boldsymbol{A}$$

$$x_i^{(\nu)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1,\, j\neq i}^{n} a_{ij} x_j^{(\nu-1)}\right) \quad , \quad i = 1,\dots,n \quad , \quad \nu = 1,2,\dots$$

nach meinen umfangreichen numerischen Experimenten kann man das Jacobi-Verfahren für einen ernsthaften Einsatz im Rahmen von Finite Elemente Analysen wegen der sehr langsamen Konvergenz oder oftmals Divergenz vergessen!

Iterationsverfahren IX

Das Gauß-Seidel-Verfahren (1874) , Einzelschritt-Verfahren, ***N*** wie folgt gewählt:

$$\boldsymbol{N} = \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ \dots & a_{22} & 0 & 0 \\ \dots & \dots & \dots & 0 \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \boldsymbol{D} - \boldsymbol{E}$$

$$\boldsymbol{M} = \boldsymbol{I} - (\boldsymbol{D} - \boldsymbol{E})^{-1} \boldsymbol{A}$$

$$x_i^{(\nu)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(\nu)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(\nu-1)} \right), \; i = 1, \dots, n, \; \nu = 1, 2, \dots$$

nach meinen umfangreichen numerischen Experimenten kann man das Gauß-Seidel-Verfahren für einen ernsthaften Einsatz im Rahmen von Finite Elemente Analysen wegen der sehr langsamen Konvergenz oder oftmals Divergenz vergessen!

Iterationsverfahren X

Das SOR-Verfahren, ***N*** wie folgt gewählt:

$$\boldsymbol{N} = \begin{pmatrix} \frac{a_{1,1}}{\omega} & 0 & 0 & 0 \\ a_{2,1} & \frac{a_{2,2}}{\omega} & 0 & 0 \\ \dots & \dots & \dots & 0 \\ a_{n,1} & a_{n,2} & \dots & \frac{a_{n,n}}{\omega} \end{pmatrix}$$

$$\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F} \quad , \quad \boldsymbol{F} = \boldsymbol{E}^T$$

$$\boldsymbol{M} = (\boldsymbol{D} - \omega \boldsymbol{E})^{-1}[(1-\omega)\boldsymbol{D} + \omega \boldsymbol{F}]$$

$$\boldsymbol{d} = -\omega(\boldsymbol{D} - \omega \boldsymbol{E})^{-1}\boldsymbol{b}$$

$$x_i^{(\nu)} = \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(\nu)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(\nu-1)} + \frac{1-\omega}{\omega} a_{ii} x_i^{(\nu-1)}\right)$$

mit $i = 1,\dots,n$, $\nu = 1,2,\dots$

Iterationsverfahren XI

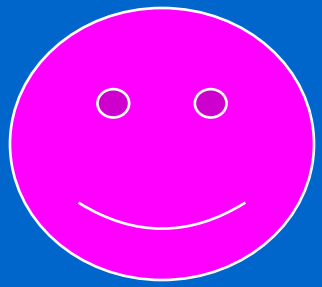
SOR-Verfahren, successive over relaxation, Relaxationsparameter

$\omega < 1$ Unterrelaxation

$\omega > 1$ Überrelaxation

Satz von Kahan: Konvergenz für $0 < \omega < 2$

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(\boldsymbol{M})^2}}$$



Diese Bestimmung ist derart aufwendig, daß man davon in der Praxis eher weniger Gebrauch macht.

Es gibt für spezielle mathematische Probleme Abschätzungen, aber für FE-Rechnungen wird man um Experimente nicht umhin kommen. Probieren Sie Werte von 0,8 bis 1,2 für den Relaxations-Parameter.

Iterationsverfahren XII

JOR-Verfahren, Jacobi over relaxation, Relaxationsparameter

$$x_i^{(\nu)} = (1-\omega)x_i^{(\nu-1)} - \frac{\omega}{a_{ii}}\left(\sum_{j=1,\, j\neq i}^{n} a_{ij}x_j^{(\nu-1)} + b_i\right),\ i = 1,\ldots,n,\ \nu = 1,2,\ldots$$

Beide, SOR- und JOR-Verfahren, enthalten als Spezialfälle mit ω=1 das Gauß-Seidel- bzw. das Jacobi-Verfahren. Aber auch diese beiden konvergenz-beschleunigten Verfahren sind nach meinen Beobachtungen nicht für übliche, größere FE- Strukturen geeignet.

Iterationsverfahren 13

Verfahren der Konjugierten Gradienten, CG-Verfahren

die Lösung ***x*** von $\boldsymbol{Ax}+\boldsymbol{b}=\boldsymbol{0}$ mit symmetrischer und positiv definiter Matrix ***A*** ist das Minimum der quadratischen Funktion

$$F(\boldsymbol{v}) := \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} v_i v_k + \sum_{i=1}^{n} b_i v_i = \frac{1}{2}(\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}) + (\boldsymbol{b}, \boldsymbol{v})$$

Der Gradient von F(***v***) ist gleich dem Residuenvektor ***r*** zum Vektor ***v***:

$$\operatorname{grad} F(\boldsymbol{v}) = \boldsymbol{A}\boldsymbol{v} + \boldsymbol{b} = \boldsymbol{r}$$

Iterationsverfahren 14

Start : Wahl von $\boldsymbol{x}^{(0)}$, bei FEA meist $\boldsymbol{x}^{(0)} = \boldsymbol{0}$

$$\boldsymbol{r}^{(0)} = \boldsymbol{A}\,\boldsymbol{x}^{(0)} + \boldsymbol{b}$$

$$\boldsymbol{p}^{(1)} = -\boldsymbol{r}^{(0)}$$

Iteration $(k = 1,2,3...)$

falls $k > 1$: $e_{k-1} = \dfrac{\boldsymbol{r}^{(k-1)^T}\boldsymbol{r}^{(k-1)}}{\boldsymbol{r}^{(k-2)^T}\boldsymbol{r}^{(k-2)}}$, $\boldsymbol{p}^{(k)} = -\boldsymbol{r}^{(k-1)} + e_{k-1}\,\boldsymbol{p}^{(k-1)}$

$$\boldsymbol{z} = \boldsymbol{A}\,\boldsymbol{p}^{(k)}$$

$$q_k = \frac{\boldsymbol{r}^{(k-1)^T}\boldsymbol{r}^{(k-1)}}{\boldsymbol{p}^{(k)^T}\boldsymbol{z}}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + q_k\,\boldsymbol{p}^{(k)}$$

$$\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} + q_k\,\boldsymbol{z}$$

Test auf Konvergenz

x Lösungsvektor
r Residummvektor
p Richtungsvektor

nach H.R.Schwarz

Iterationsverfahren 15

Zum Thema der Konvergenz läßt sich folgendes sagen, vgl. H.R.Schwarz:

Die Methode der Konjugierten Gradienten liefert die Lösung eines Gleichungssystems in n Unbekannten nach höchstens n Schritten.

Das hat eine ganz praktische Konsequenz: Bei Z88 können Sie die maximale Anzahl der Iterationsschritte wählen. Wenn Ihre zu untersuchende Struktur 100.000 Freiheitsgrade hat, dann dürfte nach höchstens 100.000 Iterationsschritten die Lösung erreicht sein.

Aber in der Praxis geht das meist erheblich schneller: Man beschleunigt das Verfahren durch eine Vorkonditionierung.

Iterationsverfahren 16

Das Verfahren der Konjugierten Gradienten mit Vorkonditionierung

$$\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}=\boldsymbol{0} \quad \Rightarrow \quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}+\tilde{\boldsymbol{b}}=\boldsymbol{0}$$

Ursprüngliches Gleichungssystem. Wir definieren eine reguläre Matrix C und bringen das Gleichungssystem auf diese Form:

$$\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{C}^{T}\boldsymbol{x}+\boldsymbol{C}^{-}\boldsymbol{b}=\boldsymbol{0}$$

Neue Größen: $\tilde{\boldsymbol{A}}=\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T} \quad , \quad \tilde{\boldsymbol{x}}=\boldsymbol{C}^{T}\boldsymbol{x} \quad , \quad \tilde{\boldsymbol{b}}=\boldsymbol{C}^{-1}\boldsymbol{b}$

Iterationsverfahren 17

Der eigentliche Trick ist nun, daß die Matrix ***C*** so beschaffen sein soll, daß gilt:

$$\kappa(\tilde{\boldsymbol{A}}) \quad < \quad \kappa(\boldsymbol{A})$$

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|$$

die Konditionszahl gilt für eine beliebige Matrixnorm

$$\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{C}^{T}\boldsymbol{x} + \boldsymbol{C}^{-}\boldsymbol{b} = \boldsymbol{0}$$

$$\boldsymbol{C}^{-T}\tilde{\boldsymbol{A}}\boldsymbol{C}^{T} = \boldsymbol{C}^{-T}\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{C}^{T} = (\boldsymbol{C}\boldsymbol{C}^{T})^{-1}\boldsymbol{A}$$

$$\boldsymbol{M} := \boldsymbol{C}\boldsymbol{C}^{T}$$

Iterationsverfahren 18

Damit wird der vorkonditionierte Algorithmus der Konjugierten Gradienten:

$$\text{Start}: \text{Wahl von } \boldsymbol{x}^{(0)} \quad , \quad \text{bei FEA meist } \boldsymbol{x}^{(0)} = \boldsymbol{0}$$

$$\text{Wahl von } \boldsymbol{M}$$

$$\boldsymbol{r}^{(0)} = \boldsymbol{A}\,\boldsymbol{x}^{(0)} + \boldsymbol{b}$$

Iterationsverfahren 19

Iteration bzw. Relaxationsschritt ($k = 1,2,3...$)

$\boldsymbol{M}\boldsymbol{\rho}^{(k-1)} = \boldsymbol{r}^{(k-1)}$, dieses Gleichungssystem muß in jedem Iterationsschritt nach $\boldsymbol{\rho}$ aufgelöst werden!

falls $k = 1$: $\boldsymbol{g}^{(k)} = -\boldsymbol{\rho}^{(k-1)}$

falls $k > 1$: $e_{k-1} = \dfrac{\boldsymbol{r}^{(k-1)^T}\boldsymbol{\rho}^{(k-1)}}{\boldsymbol{r}^{(k-2)^T}\boldsymbol{\rho}^{(k-2)}}$, $\boldsymbol{g}^{(k)} = -\boldsymbol{\rho}^{(k-1)} + e_{k-1}\,\boldsymbol{g}^{(k-1)}$

$$q_k = \frac{\boldsymbol{r}^{(k-1)^T}\boldsymbol{\rho}^{(k-1)}}{\boldsymbol{g}^{(k)^T}(\boldsymbol{A}\boldsymbol{g}^{(k)})}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + q_k\,\boldsymbol{g}^{(k)}$$

$$\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} + q_k\,(\boldsymbol{A}\boldsymbol{g}^{(k)})$$

Test auf Konvergenz

nach H.R.Schwarz

UNIVERSITÄT BAYREUTH

Iterationsverfahren 20

1) Vorkonditionierung mit SOR

Nach Evans und Axelsson wird eine Art „Relaxationsparameter“ ω gewählt und die Vorkonditionierungs- Matrix ***M*** wie folgt definiert:

$$\boldsymbol{M} := (\boldsymbol{I} + \omega\hat{\boldsymbol{E}})(\boldsymbol{I} + \omega\hat{\boldsymbol{F}}) \quad mit\ \boldsymbol{C} = \boldsymbol{I} + \omega\hat{\boldsymbol{E}} \quad , \quad \tilde{\boldsymbol{A}} = \hat{\boldsymbol{E}} + \boldsymbol{I} + \hat{\boldsymbol{F}}$$

Das Gleichungssystem $\boldsymbol{M}\boldsymbol{\rho} = \boldsymbol{r}$ wird in 2 Schritten gelöst:

$$(\boldsymbol{I} + \omega\hat{\boldsymbol{E}})\boldsymbol{y} = \boldsymbol{r}$$

$$(\boldsymbol{I} + \omega\hat{\boldsymbol{F}})\boldsymbol{\rho} = \boldsymbol{y}$$

Obwohl dadurch der Aufwand zunächst steigt, kann sich unter günstigen Bedingungen die Anzahl der Iterationsschritte so stark verringern, daß sich der Mehraufwand ohne weiteres lohnt. Für ω=1 geht das Ganze in den normalen Algorithmus der Konjugierten Gradienten über.

Iterationsverfahren 21

1) Vorkonditionierung mit SOR

Das gerade gezeigte Verfahren wird *SORCG* oder auch *SSORCG* genannt für *(symmetric) successive over relaxation conjugated gradients*, da die Matrix sinngemäß im SOR-Verfahren auftritt, wenn auch nicht als Iterationsmatrix.

Es benötigt keinen zusätzlichen Computerspeicher gegenüber dem normalen CG- Verfahren.

Iterationsverfahren 22

2) Vorkonditionierung mit partieller Cholesky-Zerlegung:

Hier nimmt man für die Gewinnung der Vorkonditionierungs- Matrix eine Cholesky-Zerlegung der Matrix vor, bei der aber jegliches *fill-in* unterdrückt wird: Alle Typen von Dreiecks- Zerlegung haben es ja so an sich, daß an Stellen, die zuvor keine Matrixelemente aufwiesen, nach Zerlegungsoperationen an solchen Stellen auf einmal Matrixelemente auftauchen. Die Elemente, die mit + gekennzeichnet sind, fallen also unter den Tisch. Damit ist die Zerlegung nur eine Approximation an , aber es wird der strukturelle Aufbau beibehalten.

x	x		x	x			
x	x	x			x		
	x	x		x			
x			x			x	x
x		x		x			
	x				x		x
			x			x	
			x		x		x

x	x		x	x			
x	x	x	+	+	x		
	x	x		x			
x	+		x	+		x	x
x	+	x	+	x		+	
	x				x		x
			x	+		x	+
			x		x	+	x

Iterationsverfahren 23

2) Vorkonditionierung mit partieller Cholesky-Zerlegung:

Die Cholesky-Zerlegung muß nicht unbedingt existieren, weil durch das fehlende *fill-in* die Matrix indefinit werden kann. Um dann auch unter solchen ungünstigen Bedingungen eine gute Vorkonditionierungs- Matrix zu bekommen, werden nach *Manteuffel* die Nicht- Diagonalelemente der skalierten Matrix ***A*** mit einem gemeinsamen Faktor, dem Shift-Faktor α, reduziert, und zwar wie folgt:

$$\tilde{\boldsymbol{A}} = \boldsymbol{I} + \frac{1}{1+\alpha}(\hat{\boldsymbol{E}} + \hat{\boldsymbol{F}})$$

Iterationsverfahren 24

2) Vorkonditionierung mit partieller Cholesky-Zerlegung:

Damit nennt man das ganze Verfahren *SICCG, shifted incomplete Cholesky conjugate gradients*. Vgl. dazu auch *Meijerink* und *van der Vorst*.

Der Shift-Faktor sollte möglichst klein sein, denn je größer er wird, umso kleiner werden die Nicht- Diagonalelemente und entfernen sich weiter von der Ursprungsmatrix. Programmtechnisch löst man das so, daß man zwar zunächst ein α einliest und startet. Funktioniert die partielle Cholesky-Zerlegung nicht, dann wird im Programm automatisch erhöht, z.B. verdoppelt.

Nachteilig an dem SICCG-Verfahren ist, daß zusätzlich Speicherplatz für die Matrix ***C*** vorgehalten werden muß. Nach meinen Beobachtungen braucht das SORCG-Verfahren etwa nur 2/3 des Speichers gegenüber dem SICCG-Verfahren.

Iterationsverfahren 25

Aber auf etwas sehr wesentliches will ich Sie noch hinweisen:

Es gibt kein 100% „richtiges“ Abbruchkriterium für Iterationssolver: Egal, welche Norm eines Residuenvektors auch immer gegen diese Schranke verglichen würde - Sie könnten niemals sicher sein, daß damit alle Elemente des Lösungsvektors richtig sind.

Dazu ein kleines Gedankenexperiment: Ein Solver A liefert 5 Werte, ein Solver B auch 5 Werte:

$$\boldsymbol{u}_A = \begin{pmatrix} 0{,}95 \\ 0{,}95 \\ 0{,}95 \\ 0{,}95 \\ 0{,}95 \end{pmatrix} \qquad \boldsymbol{u}_B = \begin{pmatrix} 1{,}05 \\ 1{,}05 \\ 1{,}05 \\ 0{,}80 \\ 1{,}05 \end{pmatrix} \qquad \boldsymbol{u}_{exakt} = \begin{pmatrix} 1{,}0 \\ 1{,}0 \\ 1{,}0 \\ 1{,}0 \\ 1{,}0 \end{pmatrix}$$

Iterationsverfahren 26

$$\boldsymbol{u}_A = \begin{pmatrix} 0{,}95 \\ 0{,}95 \\ 0{,}95 \\ 0{,}95 \\ 0{,}95 \end{pmatrix} \qquad \boldsymbol{u}_B = \begin{pmatrix} 1{,}05 \\ 1{,}05 \\ 1{,}05 \\ 0{,}80 \\ 1{,}05 \end{pmatrix} \qquad \boldsymbol{u}_{exakt} = \begin{pmatrix} 1{,}0 \\ 1{,}0 \\ 1{,}0 \\ 1{,}0 \\ 1{,}0 \end{pmatrix}$$

“ für alle drei Vektoren die euklidische

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{k=1}^{n} x_k^2}$$

dann erhalten wir folgende Werte:

$$\|\boldsymbol{u}_A\|_2 = 2{,}12 \quad , \quad \|\boldsymbol{u}_B\|_2 = 2{,}25 \quad , \quad \|\boldsymbol{u}_{exakt}\|_2 = 2{,}24$$

Auf den ersten Blick sieht es aufgrund der euklidischen Norm so aus, als wäre die Lösung B ganz dicht dran. In Wirklichkeit hält Lösung B aber eine echte Teufelei bereit: Fast alle Komponenten sind nahe an den genauen Werten, aber es gibt wenige (hier nur einen), dafür aber kapitale Ausreißer, die bestimmte Freiheitsgrade nur sehr ungenau abbilden. Hier ist Lösung A viel ausgewogener und im Endeffekt wesentlich praxisbrauchbarer.

Iterationsverfahren 27

Beachten Sie: Wenn Sie große FEA-Strukturen mit verschiedenen Solvern rechnen, wissen Sie ohnehin niemals, welcher Solver eigentlich am nächsten dran ist!

Dagegen gibt es nur ein - und wirklich nur ein - Gegenmittel: Ihr Ingenieurverstand muß Ihnen anhand der Verformungsbilder und Spannungsbilder sagen, ob diese Ergebnisse realistisch sein *könnten*. Wohl dem, der hier über sehr solide Konstruktions- und Mechanikkenntnisse, gesunden Menschenverstand und Praxis-Erfahrung verfügt!

Ach so ist das!

Spannungen und Dehnungen I

$$\boldsymbol{\sigma} = \boldsymbol{C}\,\boldsymbol{\varepsilon}$$

Das Hooke'sche Gesetz in symbolischer Matrixform:

$$\varepsilon_{ij} = \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{pmatrix}$$

Die Dehnungen für den ebenen Fall aus der Elastizitätstheorie:

$$\begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix} = \begin{pmatrix} N_1 \cdot u_1 + N_2 \cdot u_2 + N_3 \cdot u_3 + N_4 \cdot u_4 \\ N_1 \cdot v_1 + N_2 \cdot v_2 + N_3 \cdot v_3 + N_4 \cdot v_4 \end{pmatrix}$$

Das Verschiebungsfeld wird (hier für ein 4-Knoten Element) durch die Verschiebungen der Knoten mit den Formfunktionen beschrieben.

UNIVERSITÄT BAYREUTH

Spannungen und Dehnungen II

$$\varepsilon_{ij} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix} \begin{pmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix}$$

$$\boldsymbol{\varepsilon} = \boldsymbol{L}\,\boldsymbol{N}\,\boldsymbol{U}_i \quad , \quad \boldsymbol{L}\,\boldsymbol{N} = \boldsymbol{B}$$

$$\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{U}_i$$

$$\boldsymbol{\sigma} = \boldsymbol{C}\,\boldsymbol{B}\boldsymbol{U}_i$$

Spannungen und Dehnungen III

Sehr wichtig ist die Frage nach dem Ort der Spannungsberechnung. Naheliegend wäre zunächst, die Spannungen an den Knoten zu berechnen. Das ist auch ohne weiteres möglich, aber nicht sauber und eindeutig, weil die Spannungen ja elementweise berechnet werden und normalerweise mehrere Elemente an einen Knoten anstoßen. Unter Umständen muß in geeigneter Weise gemittelt werden. Bei primitiven Elementen, z.B. Scheibendreiecken mit 3 Knoten, also linearer Ansatz, ist die Spannungsberechnung ohnehin ungenau. Hier wird man die Spannungen in den 3 Knoten berechnen und damit einen Mittelwert je Element bilden.

Bei hochwertigen Elementen muß man präzise vorgehen: Die einzig saubere Methode der Spannungsberechnung in krummlinig berandeten Finiten Elementen kann nur die Berechnung der Spannungen in den Gaußpunkten sein. Das sind sozusagen die „natürlichen“ Stützstellen im Element, denn sie wurden schon während der Verschiebungsrechnung für die numerische Integration verwendet.

Spannungen und Dehnungen IV

Allerdings kann man ohne weiteres unterschiedliche Stützpunkte für Integration und Spannungsberechnung vorsehen, z.B. zum Integrieren bei der Bestimmung der jeweiligen Elementsteifigkeits-Matrix 3 x 3 Gaußpunkte und für die Punkte der Spannungsberechnung dann 4 x 4 Gaußpunkte oder sogar nur einen einzigen Gaußpunkt (d.h. den in der Mitte des Elements).

Wenn man die Spannungsplots der großen kommerziellen FEA-Programme betrachtet, dann fallen dem Betrachter meist sehr ästhetische Farbübergänge über der ganzen FE- Struktur auf und wirklich jede Stelle der Struktur weist offensichtlich eine Spannung auf. In der Realität ist das natürlich so, aber die Finite Elemente Analyse berechnet alle Werte *diskret* an Punkten und nicht kontinuierlich, wie das die Differentialgleichungen der Technischen Mechanik tun. Wie kann das also sein, wenn man die Spannungen in Wirklichkeit nur punktweise berechnen kann, entweder in den Knoten (nicht so gut) oder in den Gaußpunkten (vernünftig)?

Spannungen und Dehnungen V

Antwort: Überhaupt nicht! Hier werden einfach durch mehr oder weniger gute Interpolations- oder Approximationsfunktionen Mittelwerte aus mehreren punktweise gerechneten Spannungen gebildet und dem Betrachter als exakte Mathematik verkauft. Sieht gut aus, verkauft sich gut und wird gerne geglaubt. Mir sind allerdings mehrere Fälle von FEA-Berechnungen bekannt, bei denen sehr namhafte kommerzielle Programme definitiv *sehr falsche* Ergebnisse für die Spannungen „interpoliert“ haben, während die Verschiebungen noch richtig berechnet wurden (sic!).

Was macht Z88?
Z88O, 1. Möglichkeit: Vergleichsspannungen in den Eckknoten.

In Wirklichkeit werden die Spannungen eben nicht in den Eckknoten berechnet, was besonders bei sehr spitzen Elementen zu ganz falschen Ergebnissen führt (sic!), sondern in Gaußpunkten, die in der Nähe der jeweiligen Eckknoten liegen.

Spannungen und Dehnungen VI

Es werden die Spannungen für genau so viele Gaußpunkte berechnet, wie Eckknoten vorhanden sind. Da meist an einen Knoten mehrere Elemente anschließen, werden diese Spannungen dann noch gemittelt über alle anschließenden Elemente. Dies führt zu recht ausgeglichenen Spannungsverläufen, die im Mittel aber geringer sind als die Maximalspannungen.

Z88O, 2. Möglichkeit: Vergleichsspannungen pro Element gemittelt.

Es werden die Spannungen in den Gaußpunkten im jeweiligen Element berechnet, aufsummiert und durch die Anzahl der Gaußpunkte geteilt. Dies ergibt eine mittlere Vergleichsspannung pro Element.

Spannungen und Dehnungen VII

Z88O, 3. Möglichkeit: Vergleichsspannungen in den Gausspunkten.

SHEX88.C

```
for(j = 1;j <= 60;j++)
            {
            eps[k]= eps[k] + b[(k-1)*60 + j] * ul[j];
            }
```