National Conference with International Participation
ENGINEERING MECHANICS 2007
Svratka, Czech Republic, May 14 – 17, 2007

SPARSE DIRECT SOLVER WITH FILL-IN OPTIMALIZATION

P. Pařík*

Summary: *A symmetric sparse direct solver for finite element problems is presented. Solution of large systems can be very memory-demanding and time-consuming process, and it may even be practically impossible if a classical frontal solver is used. The presented solver is based on* **LDL**^{T} *factorization and uses various facilities to optimize solution calculations, in particular, optimal numbering scheme and efficient sparse matrix storage scheme. The solver is designed to work as an integral part of the computational system PMD.*

1. Úvod

Klasickou metodou řešení úloh metodou konečných prvků (MKP), tj. systému lineárních algebraických rovnic, je tzv. *frontální metoda* (Irons, 1970). Jedná se o variantu Gaussovy eliminace, při níž eliminace probíhá pouze na lokálních maticích elementů, které jsou ve stejné frontě, tj. kde jsou dílčí neznámé spolu „vzájemně svázány“. Eliminace nepostupuje od první do poslední rovnice, ale napřeskáčku podle toho, která neznámá již nemá žádné příspěvky od ostatních. Frontální metoda má relativně malé paměťové nároky, protože není třeba sestavovat globální matici tuhosti a eliminované rovnice se ukládají rovnou na disk. Efektivita metody (a délka fronty) závisí pouze na číslování prvků; číslování uzlů, šířka pásu nebo obrysu zde nehrají žádnou roli. Frontální metoda je výhodná pro výpočty na počítačích s malou operační pamětí, avšak je obecně pomalá a pro rozsáhlé úlohy může být v závislosti na implementaci prakticky nepoužitelná.

Kapacita paměti současných počítačů (řádově jednotky GiB) je dostatečná, aby bylo – při vhodně zvoleném způsobu uložení – možné zpracovávat celý systém rovnic (řádově i 10^7) přímo v paměti. Řešení je pak možné provést standarní metodou, např. **LDL**^{T} dekompozicí matice soustavy. Je však nezbytně nutné provést optimalizaci pořadí rovnic tak, aby se snížilo zaplnění (fill-in), které neodvratně při faktorizaci (eliminaci) vzniká. Globální matice tuhosti, sestavená z lokálních matic tuhosti jednotlivých prvků, je symetrická, řídká a má blokovou strukturu. Těchto vlastností je možné využít k efektivnímu uložení po blocích (submaticích), místo uložení jednotlivých sloupců (řádků) matice ve formě pásu nebo obrysu, a ke zjednodušení optimalizace číslování rovnic (neznámých). Protože rovnice pro výpočet reakcí je možné uchovávat přímo v globální matici tuhosti, není během faktorizace třeba žádný přístup na disk, který je řádově pomalejší než paměť.

V následujících kapitolách jsou popsány použité metody, jejich modifikace a implementace pro účely této práce. Na závěr je provedeno shrnutí současného stavu práce.

* Ing. Petr Pařík: Ústav termomechaniky, Akademie věd ČR; Dolejškova 5; 182 00 Praha 8; tel: +420 2 6605 3441; e-mail: parik@it.cas.cz

2. Popis implementace

Řešič je vyvíjen jako součást výpočtového systému PMD, který byl vytvořen již v osmdesátých letech současnými pracovníky Ústavu termomechaniky AV ČR a je používán pro širokou škálu výpočtů metodou konečných prvků. Systém PMD je kompletně napsán v programovacím jazyce FORTRAN 77 a skládá se z několika samostatných programů, z nichž každý vykonává určitou část MKP výpočtu, dle typu úlohy. Programy mezi sebou komunikují prostřednictvím standardizovaných výstupních souborů, takže do řešení lze jednoduše vkládat další programy pro různé speciální výpočty. V tom se PMD liší od komerčních „černých krabiček".

Při realizaci řešiče bylo třeba přihlédnout k určitým omezením FORTRANu 77, např. v oblasti dynamické alokace paměti, a přizpůsobit tomu i použité algoritmy. Dále jsou vyloženy hlavní metody a popsány jejich modifikace z teoretického i praktického hlediska.

2.1. Optimální číslování uzlů (*minimum degree algorithm*)

Algoritmus byl poprvé uveden v (Tinney et al., 1967) pod názvem S2. Princip je založen na symbolické analýze struktury symetrické matice soustavy, při níž se hledá takové pořadí pivotů (vyjádřené permutační maticí **P**), které při numerické faktorizaci produkuje minimální zaplnění. Místo původní matice **A** se tedy faktorizuje matice $\mathbf{PAP}^T$.

(Rose, 1970) vytvořil pro algoritmus S2 model za použití teorie grafů a přejmenoval jej na *minimum degree algorithm*. Nenulové prvky symetrické matice **A** řádu n mohou být vyjádřeny grafem $G^0=(V^0, E^0)$ s vrcholy (uzly) $V^0=\{1,\ldots,n\}$ a hranami E^0 (hrana (i,j) je v E^0 jen tehdy, když $a_{ij}\neq 0$). Protože je matice **A** symetrická, je graf G^0 neorientovaný. Tento model se nazývá *eliminační graf*.

V každém kroku faktorizace k je z V^{k-1} vybrán jeden vrchol p jako pivot. Do E^{k-1} jsou přidány další hrany tak, aby všechny vrcholy sousedící (spojené hranou) s vrcholem p byly sousedící také navzájem (tj. úplně propojený podgraf, angl. *clique*). Nakonec je z G^{k-1} odebrán vrchol p a všechny jeho přilehlé hrany a dostáváme nový eliminační graf G^k. Toto přidávání nových hran znamená, že paměťové nároky algoritmu nelze určit předem. Hrany přidané do grafu reprezentují zaplnění matice při k-tém kroku faktorizace.

Nechť $\mathrm{Adj}_{G^k}(i)$ je množina vrcholů sousedících s vrcholem i v grafu G^k. *Minimum degree algorithm* vybírá jako k-tý pivot ten vrchol p, jehož stupeň $t_p\equiv|\mathrm{Adj}_{G^{k-1}}(p)|$ je minimální ($|\ldots|$ označuje velikost množiny, tj. počet prvků).

Problém přidávání hran byl odstraněn zavedením tzv. *kvocienčního grafu* (George et al. 1980, 1989). Tento graf narozdíl od předchozího eliminačního grafu nikdy nepotřebuje více paměti než původní graf G^0.

Kvocienční graf $\bar{G}^k=(V^k, \bar{V}^k, E^k, \bar{E}^k)$ implicitně reprezentuje eliminační graf G^k, kde $\bar{G}^0=G^0$. Pro větší přehlednost je v následujícím textu vypuštěn index k. Vrcholy v $\bar{G}$ se skládají z *proměnných* V a *prvků* $\bar{V}$. Proměnné odpovídají vrcholům z eliminačního grafu G dosud neodebraným a prvky reprezentují vrcholy z eliminačního grafu již odebrané. Hrany jsou dvojího druhu: mezi proměnnými $E\subseteq V\times V$ a mezi proměnnými a prvky $\bar{E}\subseteq V\times\bar{V}$. Mezi prvky neexistují žádné hrany, neboť jsou odstraněny při tzv. pohlcování prvků. Množiny $\bar{V}^0$ a $\bar{E}^0$ jsou prázdné.

Nechť $\bar{A}_i$ je množina proměnných sousedících s proměnnou *i* v $\bar{G}$ a $\bar{E}_i$ množina prvků sousedících s proměnnou *i* v $\bar{G}$. Pro každou proměnnou *i* z množiny *V* pak platí

$$\bar{A}_i = \{ j : (i, j) \in E \} \subseteq V ,$$
$$\bar{E}_i = \{ e : (i, e) \in \bar{E} \} \subseteq \bar{V} ,$$

a

$$\mathrm{Adj}_{\bar{G}}(i) \equiv \bar{A}_i \cup \bar{E}_i \subseteq V \cup \bar{V} .$$

Nechť $\bar{L}_e$ označuje množinu proměnných sousedících s prvkem *e* v $\bar{G}$. Pro každý prvek *e* z množiny $\bar{V}$ pak platí

$$\bar{L}_e \equiv \mathrm{Adj}_{\bar{G}}(e) \equiv \{ i : (i, e) \in \bar{E} \} \subseteq V .$$

Hrany *E* a $\bar{E}$ jsou popsány množinami $\bar{A}_i$ a $\bar{E}_i$ pro každou proměnnou v $\bar{G}$ a množinami $\bar{L}_e$ pro každý prvek v $\bar{G}$. Lze ukázat, že graf $\bar{G}^k$ nezabere více paměti než původní graf $\bar{G}^0$ ($|\bar{A}^k| + |E^k| + |L^k| \leqslant |\bar{A}^0|$ pro všechna *k*).

Kvocienční graf $\bar{G}$ a eliminační graf *G* jsou spolu úzce spojeny. Je-li *i* vrcholem v *G*, je také proměnnou v $\bar{G}$ a platí

$$\mathrm{Adj}_G(i) = (\bar{A}_i \cup \coprod_{e \in \bar{E}_i} \bar{L}_e) \setminus \{ i \} . \qquad (1)$$

Když je proměnná *p* vybrána jako *k*-tý pivot, je vytvořen prvek *p* (proměnná *p* je odebrána z *V* a přidána do $\bar{V}$). Množina $\bar{L}_p = \mathrm{Adj}_G(p)$ se nalezne za použití vztahu (1), ze kterého dále vyplývá, že pro všechny prvky *e* sousedící s proměnnou *p* platí $\bar{L}_e \setminus \{ p \} \subseteq \bar{L}_p$. To znamená, že všechny proměnné, které sousedí s nějakým prvkem $e \in \bar{E}_p$, zároveň sousedí s prvkem *p*, a tudíž prvky $e \in \bar{E}_p$ již dále nejsou potřeba. Tyto prvky jsou *pohlceny* novým prvkem *p* a odstraněny; odkazy na ně jsou nahrazeny odkazem na nový prvek *p*, který je přidán do množin $\bar{E}_i$ všech proměnných *i* sousedících s prvkem *p*. Dále jsou odstraněny množiny $\bar{A}_p$ a $\bar{E}_p$, a $\bar{L}_e$ pro všechny $e \in \bar{E}_p$. Nakonec jsou z množin $\bar{A}_i$ odstraněny všechny odkazy *j*, kde jak *i*, tak *j* jsou v $\bar{L}_p$ (tj. nadbytečné hrany nahrazené sousedností s prvkem *p*).

Proměnné *i* a *j* se nazývají *nerozeznatelné* v *G*, je-li $\mathrm{Adj}_G(i) \cup \{ i \} = \mathrm{Adj}_G(j) \cup \{ j \}$. Obě proměnné jsou stejného stupně, dokud jedna z nich není vybrána jako pivot. Pokud je vybrána proměnná *i*, může být proměnná *j* vybrána jako další, aniž by vzniklo nějaké dodatečné zaplnění. Výběr *i* a *j* najednou se nazývá *hromadná eliminace*. Proměnné *i* a *j* jsou v $\bar{G}$ nahrazeny jedinou *superproměnnou*, která obsahuje zároveň *i* i *j* a je pojmenována po jedné z proměnných (označované jako *hlavní* proměnná). Proměnné, které nejsou superproměnnými, se nazývají *jednoduché proměnné*. V praxi jsou superproměnné vytvářeny v kroku *k* pouze když *i* i *j* jsou v $\bar{L}_p$. Pro určení nerozeznatelnosti se používá kvocienční graf $\bar{G}$, ve kterém platí $\mathrm{Adj}_{\bar{G}}(i) \cup \{ i \} = \mathrm{Adj}_{\bar{G}}(j) \cup \{ j \}$. Toto porovnání je rychlejší než hledání dvou nerozeznatelných proměnných v eliminačním grafu *G*, nicméně může některé případy přehlédnout (z nerozeznatelnosti v $\bar{G}$ plyne nerozeznatelnost v *G*, obrácené tvrzení však neplatí).

Množinu jednoduchých proměnných obsažených v superproměnné s hlavní proměnnou *i* označíme **i**. Když je v *k*-tém kroku jako pivot vybrána proměnná *p*, jsou eliminovány všechny proměnné v p zároveň. Použití superproměnných značně snižuje počet prováděných výpočtů stupňů proměnných, což je nejnákladnější část algoritmu.

Vnější stupeň $d_i \equiv t_i - |\mathbf{i}| + 1$ hlavní proměnné *i* se vypočítá ze vztahu

$$d_i = |\mathrm{Adj}_G(i) \setminus \mathbf{i}| = |\bar{A}_i \setminus \mathbf{i}| = |\coprod_{e \in \bar{E}_i} \bar{L}_e \setminus \mathbf{i}| \,. \tag{2}$$

kde t_i je „skutečný“ stupeň proměnné *i*. Výběr podle nejmenšího vnějšího stupně proměnné vede na lepší pořadí pivotů než výběr podle nejmenšího skutečného stupně.

Popsaný *minimum degree ordering algorithm* je založen na kvocientním grafu a zahrnuje pohlcování prvků, superproměnné a externí stupně proměnných. Detekce superproměnných se zjednodušuje použitím hash funkce na každou proměnnou, aby nebylo nutné porovnávat každou dvojici proměnných.

Approximate minimum degree

Nechť *p* je *k*-tý pivot a výpočet mezí stupňů superproměnných je prováděn pouze pro $\mathbf{i} \in \bar{L}_p$. Namísto výpočtu exaktního externího stupně d_i (2) se počítá jeho horní mez

$$\bar{d}_i^{\,k} = \min \begin{cases} n - k \\ \bar{d}_i^{\,k-1} + |\bar{L}_p \setminus \mathbf{i}| \\ |\bar{A}_i \setminus \mathbf{i}| + |\bar{L}_p \setminus \mathbf{i}| + \sum_{e \in \bar{E}_i \setminus \{p\}} |\bar{L}_e \setminus \bar{L}_p| \end{cases} . \tag{3}$$

První dva vzorce ($n-k$, řád aktivní submatice a $\bar{d}_i^{\,k-1} + |\bar{L}_p \setminus \mathbf{i}|$, zaplnění v nejhorším případě) většinou nejsou tak přísné jako vzorec třetí. Množina $\bar{L}_e$ je rozdělena na dvě disjunktní podmnožiny: *vnější* podmnožinu $\bar{L}_e \setminus \bar{L}_p$ a *vnitřní* podmnožinu $\bar{L}_e \cap \bar{L}_p$. Algoritmus při výpočtu používá pomocnou hodnotu $w(e)$, která je na začátku záporná. Je-li při procházení nalezen prvek *e*, je hodnota $w(e)$ inicializována na $|\bar{L}_e|$ a zmenšena o jedničku pro každou proměnnou *i* ve vnitřní podmnožině $\bar{L}_e \cap \bar{L}_p$. Na konci procházení prvků je $w(e) = |\bar{L}_e| - |\bar{L}_e \cap \bar{L}_p| = |\bar{L}_e \setminus \bar{L}_p|$. Pokud prvek *e* není prohledán, je $w(e)$ menší než nula. Kombinací těchto dvou případů dostaneme

$$|\bar{L}_e \setminus \bar{L}_p| = \begin{Bmatrix} w(e) & \text{pro } w(e) \geqslant 0 \\ |\bar{L}_e| & \text{jinak} \end{Bmatrix} \text{ pro všechna } e \in \bar{V} \,. \tag{4}$$

Approximate minimum degree algorithm je shodný se standardním minimum degree algoritmem s výjimkou toho, že externí stupeň proměnné d_i je nahrazen $\bar{d}_i$. Horní mez externího stupně $\bar{d}_i$ se vypočítá pomocí vztahů (3) a (4). Kromě pohlcování prvků v $\bar{E}_p$ je navíc prvkem *p* pohlcen také jakýkoliv prvek s prázdnou vnější podmnožinou $|\bar{L}_e \setminus \bar{L}_p| = 0$, a to i když s *p* přímo nesousedí. Toto „agresivní“ pohlcování elementů dále zdokonaluje výpočet mezí stupňů zmenšením $|\bar{E}|$. Podrobný rozbor algoritmu a jeho časové náročnosti je v (Davis et al., 1996).

Uvedený minimum degree algoritmus pracuje přímo s prvky matice. Jak bude popsáno v další kapitole, globální matice tuhosti má blokovou strukturu, která odpovídá incidenci uzlů v MKP síti. Toho můžeme využít a optimalizaci provést na úrovni bloků (tj. uzlů). Tento způsob je výrazně rychlejší než práce s jednotlivými prvky matice, protože optimalizaci je možné provést pouze na základě informací o topologii sítě (nehledě na řádově nižší paměťové nároky algoritmu). Důvody použití optimálního číslování uzlů jsou vysvětleny v následující kapitole.

2.1. Uložení globální matice tuhosti

V metodě konečných prvků je globální matice tuhosti sestavována z velkého počtu lokálních matic tuhosti jednotlivých prvků, které mají jednotnou strukturu. Výsledná struktura globální matice tuhosti *K* je proto bloková, kde každý blok (submatice) K_{ij} odpovídá dvojici uzlů (i, j). Jelikož v MKP síti může sousedit jen omezený počet prvků, jsou nenulové pouze ty submatice K_{ij}, jejichž příslušné uzly náleží sousedícím elementům.

Blokové uložení symetrické řídké matice je prezentováno v práci (Vondrák, 2000) pod názvem K3. Toto uložení uvažuje pouze konstantní počet stupňů volnosti ve všech uzlech sítě, proto bylo nutné vytvořit modifikované schéma, které umožňuje práci s libovolným počtem stupňů volnosti v každém uzlu.

Mějme globální matici tuhosti pro MKP síť s *N* uzly

$$\mathbf{K}=\begin{pmatrix} \mathbf{K_{11}} & \mathbf{K_{12}} & \cdots & \mathbf{K_{1N}} \\ & \mathbf{K_{22}} & & \vdots \\ & & \ddots & \vdots \\ & & & \mathbf{K_{NN}} \end{pmatrix} \tag{5}$$

a počty stupňů volnosti v jednotlivých uzlech dané vektorem

$$\mathbf{d}=\left\{d_1, d_2, \ldots, d_N\right\} . \tag{6}$$

Každá submatice K_{ij} je pak řádu $d_i \times d_j$. Obecně jsou čtvercové pouze diagonální submatice K_{ii}, není-li počet stupňů volnosti pro všechny uzly stejný.

Uložení matice v paměti

Díky symetrii a řídkosti není třeba ukládat submatice K_{ij} kde $j<i$ (tj. dolní trojúhelník) a také submatice K_{ij}, jejichž všechny prvky jsou nulové. Matice (5) je v paměti uložena jako posloupnost *bloků*, které jsou seřazeny ve spojovém seznamu. Bloky jsou ukládány po řádcích a každý řádek začíná diagonálním blokem; *i*-tý řádek odpovídá *i*-tému uzlu MKP sítě. Každý blok odpovídá nenulové matici K_{ij} a skládá se ze tří částí:

a) *pointer (adresa)* – pozice v paměti, kde začíná další blok na *i*-tém řádku nebo nula, pokud je blok na řádku poslední

b) *index* – číslo sloupce (*j*-tý sloupec odpovídá *j*-tému uzlu); v diagonálních blocích se místo toho ukládá počet mimodiagonálních nenulových bloků na příslušném řádku (tj. počet bloků spojovém seznamu)

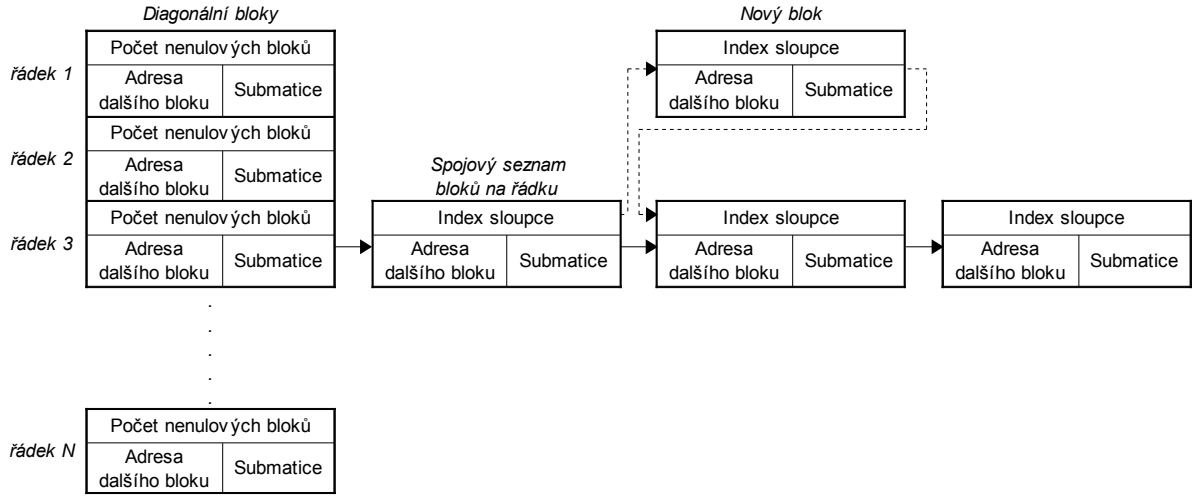
c) *submatice* – vektor prvků submatice uložený po řádcích o délce $d_i d_j$ prvků; u diagonálních submatic je uložen po řádcích pouze horní trojúhelník o délce $\left(d_i\left(d_i+1\right)\right)/2$ prvků.

Vytvoření globální matice tuhosti v paměti probíhá ve dvou krocích:

a) alokace paměti pro *N* diagonálních bloků – tuto paměť je třeba inicializovat, tj. nastavit adresy, indexy a submatice pro všechny diagonální bloky na hodnotu 0.

b) přidání uzlové submatice (i, j) – je-li $i=j$, submatice se pouze přičte do diagonálního bloku (v tomto případě není třeba alokovat žádnou další paměť, protože všechny diagonální bloky byly alokovány v kroku a). Je-li $i \neq j$, je třeba prohledat příslušný řádek *i*, počínajíc diagonálním blokem. Pokud už blok (i, j) v paměti existuje, submatice se opět pouze přičte. Pokud blok neexistuje, je třeba pro něj alokovat paměť a vložit jej na

příslušné místo v řádku, tj. zkopírovat adresu z předchozího bloku do nového bloku a nastavit adresu v předchozím bloku tak, aby ukazoval na nový blok. Při vkládání nového bloku je třeba respektovat pravidlo, že indexy (čísla sloupců) všech bloků na řádku musí být seřazeny od nejnižšího. Nakonec je třeba zvětšit index (počítadlo) v diagonálním bloku o 1.



Obr. 1 Schéma blokového uložení matice v paměti a vložení nového bloku

Paměťové požadavky na sestavení globální matice tuhosti nijak nezávisí na číslování uzlů sítě. Číslování uzlů má však rozhodující vliv na paměťové požadavky při následné faktorizaci. Nevhodně zvolené pořadí uzlů může mít za následek masivní zaplňování (fill-in) a následné zhroucení faktorizace z důvodu nedostatku paměti. Proto je nezbytně nutné před vlastním sestavením globální matice tuhosti použít vhodnou metodu optimalizace pořadí uzlů (např. minimum degree ordering, popsaný v předchozí kapitole).

3. Závěr

Byla popsána implementace přímého MKP řešiče za použití efektivního schématu uložení globální matice tuhosti a optimálního číslování uzlů pro minimalizaci zaplňování při numerické faktorizaci. Teoretické algoritmy byly přizpůsobeny pro dosažení vyšší efektivity a rychlosti výpočtů na základě analýzy struktury globální matice tuhosti a realizovány v programovacím jazyce FORTRAN. Práce je v současné době zhruba ve dvou třetinách, nicméně hlavní části programu jsou již implementovány.

Vzhledem ke stavu práce nebylo možné do článku zahrnout objektivní výsledky měření rychlosti a paměťové náročnosti řešiče, zejména v porovnání se stávajícím frontálním řešičem. Průběžné testy zatím ukazují na očekávané teoretické zvýšení rychlosti výpočtů o několik řádů.

Popsaný přímý řešič bude po dokončení sloužit k výpočtům elektronových struktur, pro které je současný frontální řešič nevyhovující. Výsledky této práce však nejsou omezeny jen na lineární úlohy; po příslušných modifikacích bude možné řešič použít i pro náročnější úlohy, které vyžadují opakovanou faktorizaci matice tuhosti v každé iteraci či v každém časovém kroku. Jedná se zejména o obecnější a efektivnější metody řešení nelineárních úloh, které jsou zatím pro praktické výpočty příliš časově nákladné.

4. Poděkování

Tato práce vzniká za podpory grantu GAV IAA100100637.

5. Literatura

Davis, T.A., Amestoy, P., Duff, I.S. (1996) An approximate minimum degree ordering algorithm. *SIAM J. Matrix Analysis and Applications*, 17, pp. 886-905.

George, A., Liu, J.W.H. (1980) A fast implementation of the minimum degree algorithm using quotient graphs. *ACM Transactions on Mathematical Software*, 6, pp. 337-358.

George, A., Liu, J.W.H. (1989) The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31, pp. 1-19.

Irons, B.M. (1970) A frontal solution program for finite element analysis. *International Journal for Numerical Methods in Engineering*, 7, pp. 5-32.

Kruis, J. (2006) *Domain Decomposition Methods for Distributed Computing*. Saxe-Coburg Publications, Glasgow.

Okrouhlík, M., Höschl, C., Plešek, J., Pták, S., Nadrchal, J. (1997) *Mechanika poddajných těles, numerická matematika a superpočítače*. Ústav termomechaniky AV ČR, Praha.

Rose, D.J. (1970) *Symmetric elimination on sparse positive definite systems and the potential flow network problem*. Ph.D. thesis, Applied Mathematics, Harvard University.

Tinney, W.F., Walker, J.W. (1967) Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proc. of the IEEE*, 55, pp. 1801-1809.

Vondrák, V. (2000) Description of the K3 sparse matrix storage system. *ODESSY Theoretical Manual*, http://www.ime.auc.dk/research/design/odessy/ods_theo.htm