© Strojniški vestnik 49(2003)11,524-537
ISSN 0039-2480
UDK 004.021:519.61/.64
Izvirni znanstveni članek (1.01)

© Journal of Mechanical Engineering 49(2003)11,524-537
ISSN 0039-2480
UDC 004.021:519.61/.64
Original scientific paper (1.01)

Hiter algoritem za poenostavljanje in obnovitev trikotniških mrež za prenos rezultatov MKE preko svetovnega spleta

A Fast Triangular-Mesh Decimation-and-Undecimation Algorithm for Transferring FEM Results via the Web

Sebastian Krivograd - Gorazd Hren - Borut Žalik - Anton Jezernik

Prispevek opisuje učinkovit algoritem za poenostavljanje grafičnega prikaza trikotniških mrež, dobljenih na primer pri analizah po metodi končnih elelementov (MKE). Dobljene mreže ohranjajo vse ključne značilnosti izvirnih mrež, pri čemer pa potrebujejo mnogo manj podatkov. Zato je metoda idealna za izmenjavo mrež prek ozkih komunikacijskih kanalov, kakršen je na primer svetovni splet. V uvodu najprej poudarjamo, da MKE, kot približna numerična metoda, običajno ustvarja količinsko izredno obsežne rezultate. V nadaljevanju podajamo kratek pregled znanih metod za poenostavljanje mrež, nato pa opišemo metodo z odstranjevanjem vozlišč. Za pospešitev iskanja najprimernejših vozlišč, ki jih je mogoče umakniti, uporabljamo sekljalno preglednico s hevristiko. Prispevek končujemo z analizo časovne in prostorske zahtevnosti ter praktičnim primerom uporabe metode pri zmanjšanju količine podatkov za prenos rezultatov po MKE prek spleta. Praktični rezultati potrjujejo teoretično časovno zahtevnost.

(Ključne besede: geometrija računalniška, poenostavljanje mrež, metode končnih elementov, svetovni splet)

This paper describes a fast algorithm for the decimation of triangular meshes, illustrated by transferring the results of a finite-element method (FEM) analysis. The obtained meshes preserve all the key characteristics of the original meshes with considerable less data, which makes the algorithm very useful for data exchange over the web. First, the FEM is briefly described as an approximate and numerical method that mostly results in an excessive quantity of data. A brief overview of the possible approaches to data reduction for triangular meshes is given, and the solution with node elimination is presented. To speed up the search for the nodes to be removed, a hash table is applied, organized heuristically and suitable for engineering data. Finally, the paper presents an analysis of a time-and-space complexity analysis and a practical example with a reduction of FEM data results, enabling efficient transfer over the web. The practical results obtained during the testing of the FEM results' transfer confirm the theoretical estimation of linear time complexity.

(Keywords: computational geometry, mesh decimation, finite element methods, world wide web)

0 UVOD

Inženirji pri svojem delu uporabljamo različne numerične metode za izvajanje analiz in preračunov komponent, kakršna je na primer metoda končnih elementov (MKE). MKE je približna numerična metoda za reševanje problemov, ki jih je mogoče opisati s sistemom diferencialnih enačb. Osnovna zamisel je razdelitev telesa na končno število elementov, omejenih z vozlišči, ki so povezani v celotni sestav z enačbami stanj in robnimi pogoji. Vrednosti v vozliščih so posledica globalnih sprememb telesa. Matrika sistema je sestavljena iz linearnih enačb, ki jih je mogoče z uporabo računalnika preprosto rešiti, je pa po obsegu zelo velika. Linearne probleme lahko rešimo v enem zagonu, pri nelinearnih pa je treba upoštevati spremembe geometrije in lastnosti materiala med obremenitvijo. Rešujemo jih iterativno dokler ni dosežen predpisani zaključni kriterij. Pri trdnostnih preračunih so rezultati izraženi kot pomiki in

0 INTRODUCTION

Many computational methods, such as the finite-element method (FEM), are well developed, implemented and used in daily engineering activities. The FEM is actually an approximate mathematical method for solving problems that can be solved with differential equations. The main idea is to break a problem into a large number of elements, determined by nodes, which are then connected via global state information and boundary conditions. The global problem is transformed as linear equations into a matrix form, which can be easily solved by a computer. The result is a change of the global state for each node. Linear problems, like structural problems, can be solved with a single solver run. Non-linear problems, where a change of the geometry of the material properties during the load application has to be con-

napetosti v posameznem vozlišču mreže elementov. V slikovni obliki so pomiki in napetosti predstavljeni z barvnimi razredi za upodabljanje porazdelitve in koncentracij napetosti in deformacij po strukturi. Računalniški prikaz rezultatov v sistemih za preračune je zelo kakovosten.

V prispevku ne bomo govorili o vedno aktualni problematiki optimalnega generiranja mrež, pač pa o možnostih prenašanja in predstavljanja rezultatov izračunov prek spleta. Pri sočasnem inženiringu oziroma sodelovanju udeležencev proizvodnega postopka se velikokrat pojavi problem prenašanja rezultatov med inženirji ali drugimi udeleženci zaradi vrednotenja rezultatov. Za izmenjavo podatkov obstaja nekaj utečenih poti:

- vsi udeleženci imajo in znajo uporabljati enak sistem za analize po MKE (kar je redko),
- mogoča je neposredna sprememba podatkov med različnimi sistemi za MKE (večinoma drago),
- obstajajo nevtralne oblike, ki jih lahko različni sistemi uvažajo ali izvažajo (izguba informacij),
- rezultate lahko izmenjujemo v obliki zaslonskih slik (zelo omejene možnosti predstavitve, posebej pri 3D analizah, nezmožnost geometrijskih preoblik, lastne nastavitve barv in pogledov).

Pri tem poudarjamo, da je izmenjava podatkov mogoča med mnogimi sistemi (model MKE z mrežo elementov in robnimi pogoji), le redko pa je mogoča izmenjava rezultatov. Vsekakor lahko trdimo, da glavni problem pri pošiljanju podatkov o modelih MKE prek spleta (še posebej pri rezultatih) pomeni količina podatkov. Celo preproste mreže MKE lahko vsebujejo nekaj tisoč vozlišč in v vsakem vozlišču je več vrednosti. Število teh vrednosti je odvisno od števila prostostnih stopenj uporabljenih elementov. Pri nelinearnih problemih se število podatkov še bistveno poveča. Prav velika količina podatkov, ki ustvarjajo rezultate, je glavna ovira pri prenosu le-teh prek računalniškega omrežja. Inženirjem so v večini primerov zanimivi dovolj natančni rezultati deformacij in zgostitev napetosti v kritičnih območjih, medtem ko so drugi podatki manj pomembni. Prav v takšnih primerih se izkaže naš algoritem, ki razumno zmanjša količino potrebnih podatkov za vizualizacijo rezultatov analize po MKE.

Poenostavljanje trikotniških mrež lahko opravimo na več načinov glede na to, katere elemente odstranjujejo iz mreže:

- **Poenostavljanje vozlišč.** Poenostavljanje vozlišč je najpogostejše in temelji na Schroederjevem poenostavitvenem postopku [1]. Vozlišča najprej ovrednotimo in jih nato odstranimo iz mreže, glede na njihovo pomembnost. Pregled metod najdemo v [2].
- **Poenostavljanje robov.** V tem primeru odstranjujemo pred tem ovrednotene robove [3]. Ena najboljših metod s poenostavljanjem robov temelji na najmanjši vsoti kvadratov napak, ki sta jo predlagala Gerland in Heckberg [4].

sidered, are solved in more than one step, mostly iteratively, using the results of the last run as the start value for the next run, until certain exit conditions are fulfilled.

In this paper we will present the sharing and presentation of results over the web. With today's concurrent engineering and the need for cooperation between production participants, problems arise when one is trying to share results with other people, in some cases other engineers, looking for comments or evaluating results. The existing possibilities for sharing results are as follows:

- Both participants have the same FEM system and know how to use it (this is a rare situation).
- Using a direct translation program between two different systems, produced by FEM systems' providers (this is usually expensive).
- Using neutral meta-files that different systems can export and import (this can result in information loss).
- Sending screen shots represents a very limited visualisation capability, especially for 3D models.

It should be pointed out that many systems can share pre-processing data (a FEM model with mesh and boundary conditions) and a few results. The major bottleneck when transferring data via the web is the quantity of data. Even a simple FEM mesh can result in a few thousand nodes and in every node we have a few values of the results, depending on the type of elements and the number of DOF in the nodes. If we consider non-linear problems, which are solved in more than one step, the quantity of data increases significantly. Usually, at the nodes, the scalar values are given. Frequently, a huge amount of obtained data slows the computer-supported analysis and increases the time and network capacity needed for data transfer. Usually, engineers are looking for sufficiently accurate results in their areas of interest and the remaining data are less significant. Therefore, these less important data can be eliminated without significantly damaging the investigation. At this point our algorithm makes an apearance and dracstically reduces the quantity of results.

Many approaches to triangular-mesh decimation have been developed. Roughly, they can be divided according to the elements they are taking from the mesh:

- **Node decimation methods** are the most frequently used and are based on the Schroeder simplification algorithm [1]. The nodes are evaluated, and they are incrementally removed from the mesh according to their importance. There are various techniques proposed, and they differ in terms of how the nodes are evaluated and what type of triangulation is required [2].
- **Edge decimation methods** eliminate edges, according to the evaluation [3]. One of the best edge decimation methods, based on a quadratic error matrix, was proposed by Garland and Heckberg [4].

- **Poenostavljanje trikotnikov.** V tem primeru naj bi algoritem odstranjeval trikotnike, vendar praktične izvedbe tega postopka niso znane.

V prispevku predstavljen algoritem temelji na dveh postopkih poenostavljanja trikotniških mrež. Franc in Skala ([5] in [6]) sta uporabila vzporedni algoritem s sekljalno preglednico, s čimer sta pospešila iskanje vozlišč, ki jih je treba odstraniti. Njuna metoda temelji na kombinaciji odstranjevanja vozlišč in robov, tako da najprej izbereta najprimernejše vozlišče in nato poiščeta med robovi, ki se stikajo v njem, najkrajšega, ki ga odstranita. Naš algoritem uporablja metodo za poenostavljanje vozlišč, kakor jo je predstavil Schroeder [1], za pospešitev pa sekljalno preglednico. Vsebuje tudi inverzno metodo, to je možnost obnovitve izvorne trikotniške mreže.

1 ALGORITEM POENOSTAVLJANJA

Algoritem za poenostavljanje trikotniških mrež sestoji iz naslednjih korakov:

1. ovrednotenje vseh vozlišč glede na izbran kriterij in njihova razvrstitev v sekljalno preglednico;
2. izbira najprimernejšega vozlišča z uporabo sekljalne preglednice (na primer vozlišče v_i na sliki 1a);
3. odstranitev vozlišča iz trikotniške mreže;
4. odstranitev vseh trikotnikov, ki so se stikali v odstranjenem vozlišču (slika 1b);
5. omreženje področja, iz katerega smo odstranili trikotnike (slika 1c);
6. ponovno ovrednotenje neposrednih sosedov odstranjenega vozlišča (vozlišča v_j, v_k, v_l, v_m, v_n na sliki 1c);
7. vračanje na korak 2, dokler ni izpolnjen končni kriterij.

1.1 Ovrednotenje vozlišč

Pred postopkom poenostavljanja moramo ovrednotiti vsa vozlišča. Ovrednotenje lahko opravimo na več načinov, na primer:

- **Triangle decimation methods** eliminate one or more triangles. However, the approaches using this possibility have not yet been reported.

The presented algorithm combines two approaches during mesh decimation. Franc and Skala ([5] and [6]) used a hash table in a parallel environment for speeding up the search for the most suitable node to be removed. They combined node and edge removal: first, the most suitable node is selected, and then from the edges incident to that node, the shortest one is contracted. Our algorithm uses the pure node decimation proposed by Schroeder [1], introducing the hash table as the acceleration technique. The undecimation in our algorithm represents an additional feature that enables an incremental reconstruction possibility for the original mesh, which makes it very suitable for engineering applications.

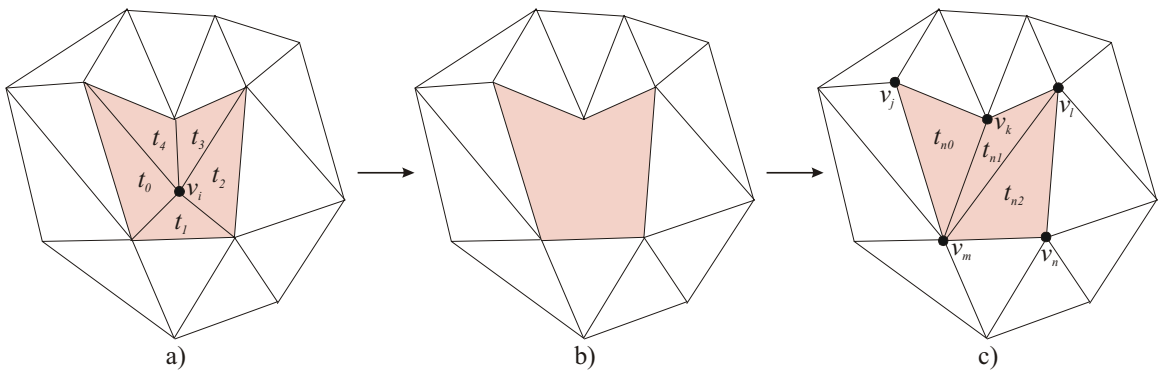
1 THE DECIMATION ALGORITHM

The algorithm for triangular-mesh decimation consists of the following steps:

1. Evaluation of all the nodes according to a chosen evaluation criterion and arranging them into a hash table.
2. Selecting the most suitable node using the hash table (for example, node v_i in Fig. 1a).
3. Removing the node from the triangular mesh.
4. Removing all the triangles incident on the removed node (Fig. 1b).
5. Triangulating the area from where the triangles have been removed (see Fig. 1c).
6. Re-evaluation of the nodes incident on the removed node (nodes v_j, v_k, v_l, v_m, v_n in Fig. 1c).
7. Returning to step 2 until the termination criterion is met.

1.1 Evaluation of nodes

Before the decimation process starts, the nodes have to be evaluated. The evaluation can be done using different criteria, for example:



Sl. 1. *Poenostavljanje vozlišč*
Fig. 1. *Node decimation*

- Ustvarimo vektor $\boldsymbol{v}_{i,j}$, ta povezuje vozlišče v_i, ki ga želimo ovrednotiti, in njegovo sosednje vozlišče v_j (slika 2a). Zatem izračunamo kot med tem vektorjem in ravnino XY ($\gamma_{i,j}$). Povprečna vrednost vseh kotov $\gamma_{i,j}$, ki so določeni z vozliščem v_i, je faktor ovrednotenja ev_i tega vozlišča.
- Faktor ovrednotenja e_{vi} vozlišča v_i je povprečna vrednost razlik skalarnih vrednosti f_i med izbranim vozliščem v_i in njegovimi sosedi (sl. 2b).

1.2 Izbira vozlišča za odstranitev

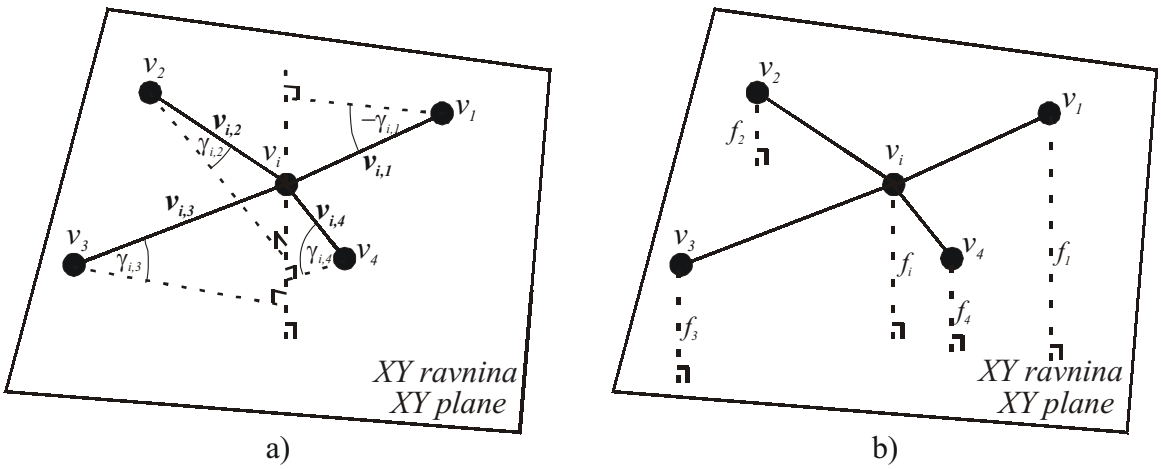
Odstranjeno vozlišče naj bi povzročilo najmanjšo napako v predstavitvi podatkov. Zaradi tega bi morali vedno odstraniti iz trikotniške mreže vozlišče z najmanjšim faktorjem ovrednotenja ev_i. Po tem opravilu se faktor ovrednotenja ev_i spremeni vsem vozliščem, ki so neposredni sosedi brisanega vozlišča, zato moramo ta vozlišča ponovno ovrednotiti. V naslednjem koraku ponovno potrebujemo vozlišče z najmanjšim faktorjem ovrednotenja ev_i. Iskanje vozlišča z najmanjšim faktorjem ovrednotenja bi lahko najlažje opravili z iskanjem skozi celotno množico vozlišč. Na žalost ta metoda deluje s časovno zahtevnostjo $O(n^2)$ in zelo upočasni algoritem. Druga možnost, da najprej uredimo vsa vozlišča glede na njihov faktor ovrednotenja ev_i in nato glede na spremenjene faktorje spreminjamo njihove položaje, deluje v pričakovani časovni zahtevnosti $O(n \log n)$. Z uporabo sekljalne preglednice dosežemo nespremenljivo časovno zahtevnost $O(1)$. V tem primeru moramo omejiti zahtevo po izbiri vozlišča, ki ima najmanjši faktor ovrednotenja ev_i. Sedaj ne zahtevamo več, da vedno uporabimo vozlišče z najmanjšim faktorjem ovrednotenja ev_i, ampak se zadovoljimo z vozliščem, ki ima dovolj majhen faktor ovrednotenja tako, kakor pojasnjujemo v nadaljevanju. Število vstopov m v sekljalno preglednico določimo z naslednjo hevristično formulo:

- Vector $\boldsymbol{v}_{i,j}$ connecting the examined node v_i and its neighbouring node v_j is formed (see Figure 2a). The angle between this vector and the *xy* plane is calculated ($\gamma_{i,j}$). The average value of all angles $\gamma_{i,j}$ defined by node v_i is used as the evaluation value ev_i.
- The average difference of the scalar values f_i between the examined node v_i and the neighbouring nodes is used as the evaluation value ev_i (Figure 2b).

1.2 Selection of a node to be removed

The removed node should cause the smallest possible change in the data representation. Therefore, the node with the smallest evaluation value ev_i should be selected and removed from the mesh. After this, the evaluation values ev_i of the neighbouring nodes have to be estimated again. For the next iteration step, the algorithm again needs the node with the smallest ev_i. Walking through the set of remaining nodes, and selecting the one with the smallest ev_i is an easy way to perform the task. Unfortunately, this method works in $O(n^2)$ time and slows down the algorithm significantly. The second possibility is first sorting all the nodes according to ev_i and then adjusting their position in the sorted array according to the changed ev_i, which works, as expected, in $O(n \log n)$ time. The constant expected time complexity can be achieved by introducing a hash table. However, the condition of selecting the node with the smallest ev_i has to be relaxed slightly. The nodes are organised in the hash table according to their evaluation values ev_i. The number of entries m into the hash table has been determined by the following heuristic:

$$m = \left\lfloor \frac{n}{k} \right\rfloor \quad k > 0 \qquad (1),$$



Sl. 2. *Ovrednotenje vozlišč*
Fig. 2. *The evaluation of nodes*

kjer k pomeni uporabniško podano stalnico. Ker je v večini tehničnih uporab faktor ovrednotenja ev_i porazdeljen po eksponentnem zakonu, izračunamo meje intervalov d_j, $0 \le j \le m$, v sekljalni preglednici z naslednjo enačbo:

where k is the constant given by the user. As in engineering applications, the distribution of the ev_i often follows the exponential law. The boundaries of the intervals d_j, $0 \le j \le m$, in the hash table are determined by the following equation:

$$d_j = ev_{sum} - ev_{\ln} \cdot \ln\left(1 + \Delta ev - j \cdot ev_{\Delta m}\right), \quad 0 \le j \le m \tag{2}.$$

Lego vozlišča v sekljalni preglednici izračunamo po enačbi:

and the position of the node in the hash table is then obtained by:

$$j = \begin{cases} \left\lfloor ev_{\Delta m_{in}} \cdot \left(1 + \Delta ev - e^{\left((ev_{sum} - ev_i) \cdot ev_{\ln_{in}}\right)}\right)\right\rfloor & ev_i < ev_{\max} \\ m-1 & ev_i \ge ev_{\max} \end{cases} \tag{3},$$

kjer so:
- ev_i: faktor ovrednotenja vozlišča i
- $ev_{\min}$: najmanjši faktor ovrednotenja ev_i
- $ev_{\max}$: največji faktor ovrednotenja ev_i
- m: število korakov, in

where:
- ev_i: evaluation value of node i,
- $ev_{\min}$: minimum value of evaluation value ev_i,
- $ev_{\max}$: maximum value of evaluation value ev_i,
- m: number of intervals, and

$$ev_{sum}: (ev_{\min} + ev_{\max})$$

$$ev_{\ln}: \frac{ev_{\max}}{\ln(1+\Delta ev)} \qquad ev_{\ln_{in}}: \frac{\ln(1+\Delta ev)}{ev_{\max}}$$

$$\Delta ev: (ev_{\max} - ev_{\min})$$

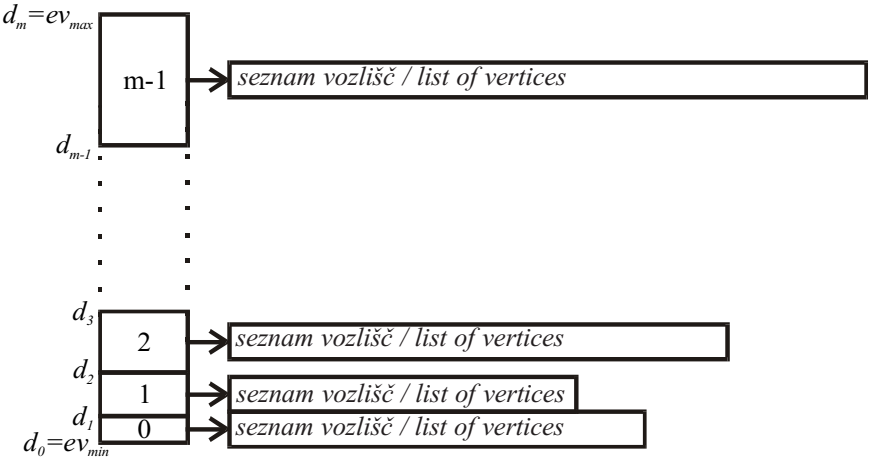
$$ev_{\Delta m}: \frac{\Delta ev}{m} \qquad ev_{\Delta m_{in}}: \frac{m}{\Delta ev}$$

Slika 3 prikazuje strukturo sekljalne preglednice. Vozlišča so v vsakem koraku shranjena v vrsto FIFO.

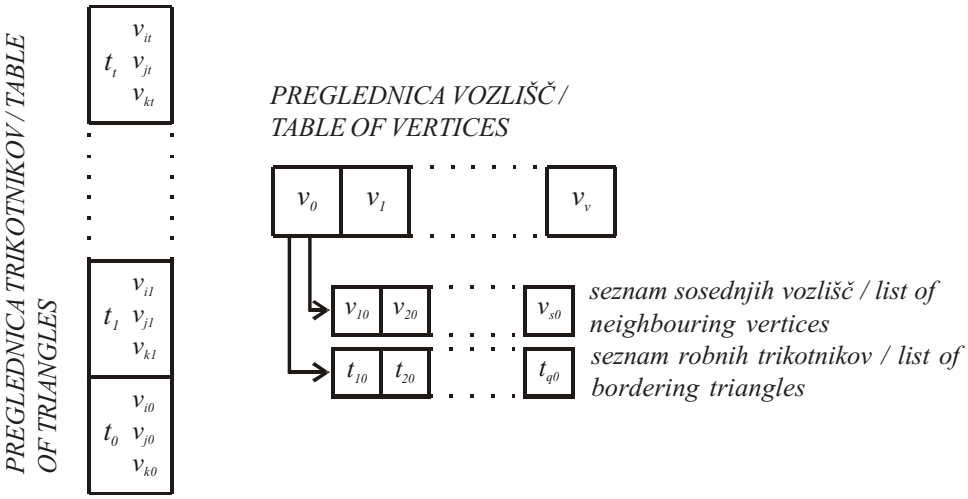
S takšno strukturo je zelo preprosto dobiti vozlišče, ki naj bo odstranjeno v naslednjem koraku. Algoritem izbere prvo vozlišče iz najnižjega polnega koraka in ga briše. Neposredne sosede je treba ponovno ovrednotiti in potem postaviti na konec vrste FIFO ustreznega koraka. Z vstavljanjem na konec vrste preprečimo, da bi se poenostavljanje izvajalo samo krajevno. Sekljalna

Figure 3 schematically shows the structure of the hash table. The nodes at each interval are stored in a FIFO queue.

With such a structure the next node to be removed from the mesh is easily obtained. The algorithm selects the first node from the lowest non-empty interval and removes it. The neighbouring nodes of the removed node are then re-evaluated. These nodes are then inserted in the corresponding interval at the end of the FIFO queue. Inserting nodes at the end of the queue prevents the decimation process from being performed



Sl. 3. *Porazdelitev vozlišč v sekljalno preglednico*
Fig. 3. *Distribution of nodes into the hash table*



Sl. 4. *Neposredni dostop do sosednjih vozlišč*
Fig. 4. *Direct access to neighbours' nodes*

preglednica zagotavlja nespremenljivo časovno zahtevnost prvega dela algoritma. S shranjevanjem sosednjih vozlišč lahko dobimo te sosede brez predhodnega iskanja (slika 4). Ker ima vsako vozlišče l sosednjih vozlišč in ker velja $l << n$, se za ponovno razporeditev sosednjih vozlišč porabi $O(l) \approx O(1)$ časa.

V primeru, da robnih vozlišč ne želimo odstraniti, jih ne vstavimo v sekljalno preglednico.

1.3 Triangulacija mnogokotnega območja

Po odstranitvi vozlišča iz trikotniške mreže se odstranijo tudi vsi trikotniki, ki se stikajo v tem vozlišču (osenčen del na sl. 1). Prazno območje moramo napolniti z novimi trikotniki. Franc in Skala [6] sta tukaj uporabila zanimivo rešitev z izbiro najkrajšega roba med brisanim vozliščem in njegovimi sosedi. Izbrani rob se skrči v sosednje vozlišče, skupaj z robovi, ki se v njegovih vozliščih stikajo. Ta eleganten postopek pa predpostavlja, da je meja področja trikotnikov, ki se stikajo v ogliščih odstranjenega roba, vedno konveksna. V praksi se na žalost izkaže, da ta predpostavka vselej ni pravilna. Zaradi tega smo raje uporabili klasični omrežilni algoritem z odstranjevanja uhljev [8].

2 OBNOVITEV TRIKOTNIŠKE MREŽE

Vračanje odstranjenih vozlišč v trikotniško mrežo v nasprotnem vrstnem redu, kakor so bila odstranjena, je zelo uporabno, saj daje uporabniku možnost določitve najugodnejšega števila odstranjenih trikotnikov. Tako lahko uporabnik po korakih rekonstruira brisana vozlišča ali pa obdela celotno množico vozlišč z različnimi kriteriji. Postopek obnavljanja lahko izvedemo zelo učinkovito s primerno podatkovno strukturo. Na začetku imamo seznam vozlišč in seznam trikotnikov. Po brisanju vozlišč iz

only locally. The hash table ensures the constant time complexity of this part of the algorithm. By storing a list of neighbouring nodes at each node of the mesh, the neighbouring nodes are accessible without any search (see Figure 4). As each node has l neighbouring nodes, in general $l << n$, the update of the estimation values is realised in $O(l) \approx O(1)$ time.

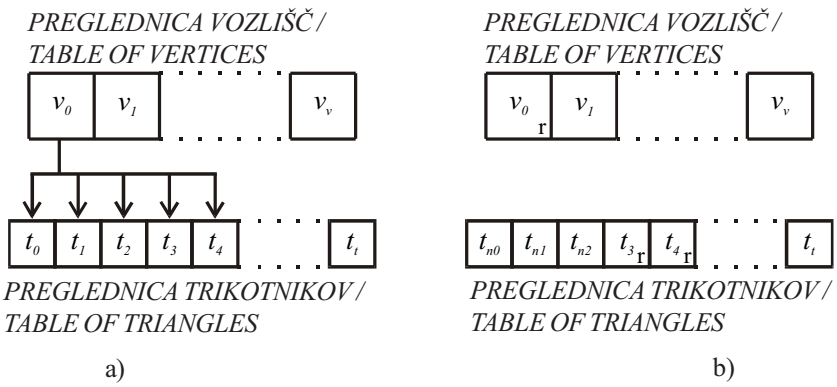
Frequently, it is desirable that the border nodes aren not included in the decimation process, this is achieved by not inserting these nodes into the hash table.

1.3 Triangulation of a polygon area

After removing a node from the mesh, all the triangles incident to it, are eliminated (shaded part in Fig. 1) and the region has to be triangulated again. Here, Franc and Skala [6] applied a solution involving a selection of the shortest edge from the removed node to their neighbours and contracted pulling all the edges defined by the removed node to the opposite node to the shortest edge. However, this elegant method works only when the obtained gap forms a convex polygon, which is, unfortunately, in practice very rare. For this reason we applied a classical polygon ear-cutting triangulation algorithm [8].

2 UNDECIMATION

Returning the removed nodes to the mesh in the reverse order of their elimination is an extremely useful feature in practice, giving the user the opportunity to experiment with the mesh. In this way, the user may return, step by step, only a few nodes instead of processing the whole set of nodes again and again, trying different termination criteria. At the beginning we have an array of nodes and an array of triangles. The position of nodes remains the same, they are just pulled-out from the mesh. The removed



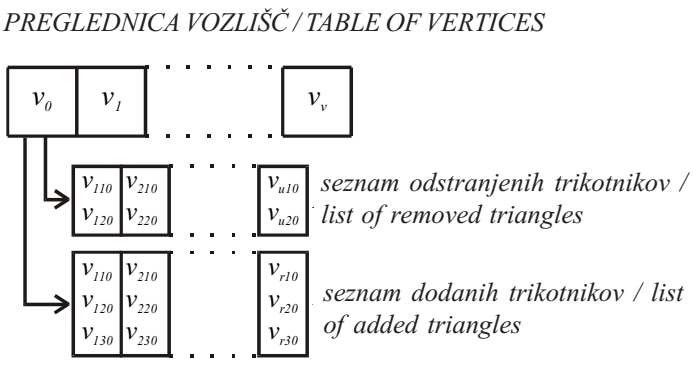
Sl. 5. *Stanje podatkovne strukture po brisanju vozlišča*
Fig. 5. *The state of the data structure after removing a node*

trikotniške mreže ostane njihov položaj v seznamu nespremenjen, označimo le, da so vozlišča brisana. Z brisanjem vozlišča se brišejo tudi trikotniki, ki se stikajo v tem vozlišču. Ker je vedno manj novih trikotnikov, kakor je bilo brisanih, le-ti zavzamejo njihova mesta v seznamu trikotnikov. Slika 5 prikazuje stanje podatkovne strukture po brisanju vozlišča v_0 v primeru, ki ga prikazuje slika 1.

Postopek obnavljanja potrebuje zapis o vsakem koraku izvajanja postopka poenostavljanja. Najlažja metoda bi bila, da bi shranjevali celotno topologijo vseh dobljenih trikotniških mrež, kar pa bi zahtevalo opravila z datotekami, s čimer bi zelo upočasnili celoten postopek. Z učinkovito organizacijo podatkovne strukture lahko ta postopek izvedemo z dovolj majhno količino dodatnega pomnilnika. Slika 6 prikazuje predlagano rešitev. Vsakemu brisanemu vozlišču dodamo dva seznama. Prvi seznam vsebuje indekse vozlišč odstranjenih trikotnikov. Potrebujemo le dva indeksa, saj pomeni tretje vozlišče brisano vozlišče. Drugi seznam vsebuje indekse vozlišč novih trikotnikov. Postopek rekonstruiranja je sedaj zelo preprost. Vozlišču, ki ga želimo vrniti v trikotniško mrežo, brišemo označbo, da je brisano. Trikotnike, ki so bili ustvarjeni pri brisanju vozlišča, odstranimo, trikotnike, ki smo jih prej odstranili, pa rekonstruiramo.

node is marked with a flag. When the node is removed, the triangles sharing that node are removed, too. There are always fewer new triangles and they occupy the memory locations of the old ones. Figure 5 shows the state of the data structure after removing node v_0 in the example shown in Fig. 1.

The undecimation process requires a knowledge of how the process of decimation was executed and what changes in the triangular mesh occur during each step. The easiest solution would be to store the topology of each mesh obtained, which involves a file operation and a slowing down of the whole process. However, with a proper organisation of the data, the shape of the mesh can be restored with a modest amount of additional memory. Figure 6 demonstrates the solution. Two additional one-way linked lists are introduced at each removed node. The first list stores the indices of the removed triangles containing two indices; the third one is the removed node itself. The second list stores the indices of the new triangles. The undecimation process is now a straightforward task. The node to be returned to the mesh sets up the flag indicating that it belongs to the mesh again. The triangles that have been added by the polygon triangulation process are removed, and the information about the old triangles is obtained from the list of the removed triangles.



Sl. 6. *Dodatna seznama pri brisanem vozlišču*
Fig. 6. *The additional lists at the removed node*

3 REZULTATI

V tem poglavju podajmo najprej teoretično analizo časovne in prostorske zahtevnosti metode, nato pa pokažemo primernost metode na trikotniški mreži analize MKE.

3.1 Časovna in prostorska zahtevnost

Predlagani algoritem za poenostavljanje trikotniških mrež sestoji iz naslednjih korakov:
- ovrednotenje vozlišč izvedemo v času *O(n)*, kjer *n* pomeni število vhodnih vozlišč;
- brisanje vozlišča v_i izvedemo v nespremenljivem času *O(*1*)*;
- omreženje mnogokotnika z uporabo algoritma odstranjevanja uhljev izvedemo v času $O(l_i^2)$, kjer l_i pomeni število sosednjih vozlišč brisanega vozlišča v_i. Ker velja $l_i << n$, lahko trdimo, da se ta korak izvede v enakem času *O(1)* glede na *n*;
- ponovno ovrednotenje sosednjih vozlišč brisanega vozlišča tudi opravimo v enakem času *O(1).*

Če je *k* število vseh vozlišč, ki so bila odstranjena med postopkom poenostavljanja, potem je ocena časovne zahtevnosti:

$$T(n) = O(n) + k \cdot \left(O(1) + O(1) + O(1)\right) = O(n) + O(k) = O(n) \quad (4).$$

Enaka časovna ocena velja tudi za postopek obnovitve.

Ostala je še ocena prostorske zahtevnosti algoritma. Na začetku dodelimo *n* zapisov za vozlišča in 2*n* zapisov trikotnikov (vsako omreženje sestoji iz največ 2*n*–*h*–2 trikotnikov, kjer je *h* število vozlišč, ki sestavljajo konveksno lupino mnogokotnika [9]). Pri vsakem brisanem vozlišču v_i potrebujemo l_i zapisov za odstranjene trikotnike in $l_i - a, 0 < a \le l_i, l_i << n$, zapisov za dodane trikotnike. Tako dobimo oceno prostorske zahtevnosti:

$$S(n) = O(n) + O(2n) + k \cdot O(1) = O(n) \quad (5).$$

3.2 Praktični rezultati

Algoritem smo testirali na veliko primerih trikotniških mrež s skalarnimi vrednostmi v vozliščih in s tem dobili veliko zanimivih in vzpodbudnih rezultatov o učinkovitosti algoritma. Predstavili bomo dve popolnoma različni množici testnih podatkov. Prvo množico sestavljajo ustvarjeni podatki, ki jih uporabimo za testiranje časovne učinkovitosti. Drugo množico pa predstavlja preprost trikotniški model MKE za prikaz učinkovitosti tega orodja.

3.2.1 Merjenje porabljenega časa

Za testiranje smo ustvarili podatke, pri katerih so bila vhodna vozlišča organizirana v

3 RESULTS

In this section, an analysis of the theoretical time-and-space complexity is performed first, this is followed by a presentation of the FEM analysis.

3.1 Theoretical time-and-space complexity

As we have seen, the proposed algorithm for mesh decimation consists of the following steps:
- the evaluation of the nodes is done in *O(n)*, where *n* is the number of input nodes,
- the removal of a node v_i is realised in constant time *O(*1*)*,
- the triangulation of a polygon using ear-cutting is performed in $O(l_i^2)$, where l_i is the number of neighbouring nodes of the removed node v_i. However, $l_i << n$, and therefore, this step can be considered as being done in constant time *O(1)* in terms of *n*.
- the re-evaluation of the neighbouring nodes of the removed node is done again in constant time *O(1).*

If *k* is the number of all the nodes that are removed during the decimation process, the required time complexity becomes:

The process of undecimation follows the same pattern.

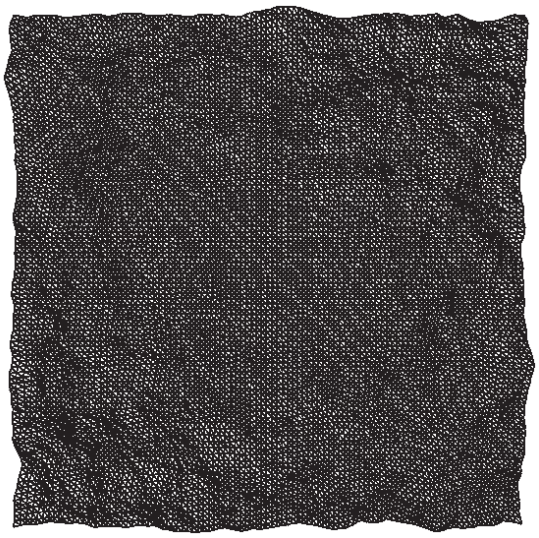
If we investigate the space complexity of the algorithm, the space for *n* nodes and 2*n* triangles is allocated at the beginning (it is well-known that each triangulation consists of at most 2*n*–*h*–2 triangles, where *h* is the number of nodes forming the convex hull of the given set of polygons *[9]*). During the removal of each node v_i, l_i records about removed triangles and $l_i - a, 0 < a \le l_i, l_i << n$, records about added triangles are needed. In this way we obtain:

3.2 Practical results

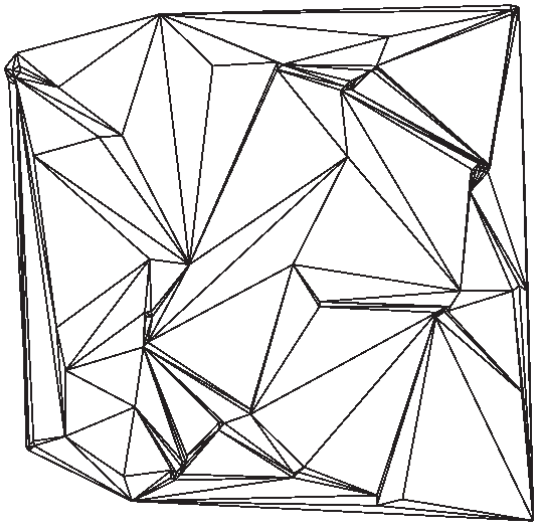
We tested the algorithm on many different sets of data consisting of meshes with scalar values at the nodes and obtained many interesting and encouraging results about algorithm efficiency. Two different testing-data sets are presented. First, an artificial data set is used for testing the time efficiency and, second, simple FEM model meshes considering the efficiency of the described tool are presented.

3.2.1 Time testing

We produced a mesh arranged as a regular grid. The example consists of 10,000 up to



Sl. 7a. *Trikotniška mreža z 10.000 vozlišči*
Fig. 7a. *Mesh with 10,000 nodes*



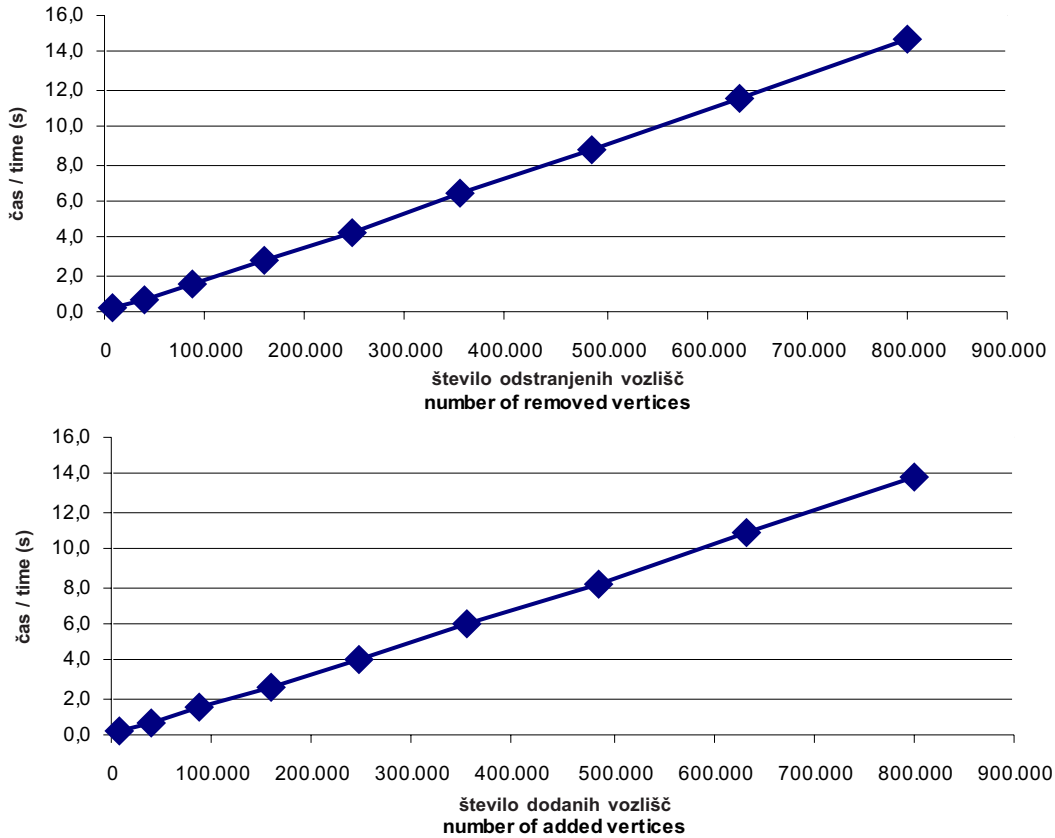
Sl. 7b. *Poenostavljena trikotniška mreža s 100 vozlišči*
Fig. 7b. *Mesh decimation to 100 nodes*

regularno mrežo. Pri procesu testiranja smo najprej poenostavili originalno trikotniško mrežo (slika 7a) z odstranitvijo 99% vozlišč (sl. 7b prikazuje dobljeno trikotniško mrežo s samo enim odstotkom vozlišč), le-to smo nato obnovili v originalno mrežo.

Preglednica 1 prikazuje porabljen procesorski čas za poenostavljanje in obnovitve

1,000,000 nodes. In the testing procedure, 99% of the nodes were first eliminated from the original mesh (Figure 7a) to obtain a mesh with 1% of nodes (Figure 7b), and then reconstructed into the starting mesh.

Table 1 shows the CPU time needed for the mesh decimation and undecimation. The tests



Sl. 8. *Grafa časov, potrebnih za poenostavitev (zgoraj) in obnovo (spodaj) trikotniške mreže*
Fig. 8. *Graphs of times needed for mesh decimation (abowe) and undecimation (below)*

Preglednica 1. *Časi, potrebni za poenostavljanje in obnovitev*
Table 1. *Times needed for mesh decimation and mesh undecimation*

VHOD INPUT	št. vozlišč (x1000) no. of nodes (x1000)	10	40	90	160	250	360	490	640	810
IZHOD OUTPUT	št. vozlišč no. of nodes	100	400	900	1600	2500	3600	4900	6400	8100
	št. trikotnikov no. of triangles	188	775	1772	3167	4959	7156	9748	12742	16121
ČAS (s) TIME (s)	poenostavljanje decimation	0,140	0,651	1,583	2,774	4,357	6,340	8,742	11,526	14,681
	rekonstrukcija undecimation	0,140	0,611	1,442	2,583	4,086	5,928	8,202	10,816	13,840

trikotniških mrež. Za testiranje smo uporabili osebni računalnik s procesorjem Celeron 600 MHz in 384 MB spomina. Linearno obnašanje algoritma lahko vidimo iz grafov, ki ju prikazuje slika 8.

3.2.2 Model MKE

Za testiranje uporabnosti algoritma pri prikazu rezultatov numeričnih metod ni nujna uporaba velikih zapletenih problemov, saj so prednosti očitne že pri preprostih primerih, učinkovitost na bolj zapletenih primerih, ki pomenijo tudi večje število podatkov, pa je še večja. Za testiranje smo uporabili rezultate trdnostnega preračuna kvadratne plošče z luknjo, ki je prikazana na sliki 9, kjer so prikazani tudi robni pogoji (porazdeljena obremenitev na zgornjem robu in stalne podpore na spodnjem) in mreža elementov. Mrežo sestavlja 30458 linearnih izometričnih trikotniških elementov, ki jih določa 15559 vozlišč.

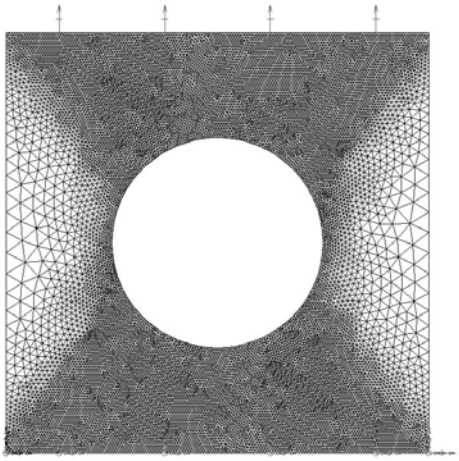
Slika 10 prikazuje rezultate v obliki porazdelitve primerjalnih napetosti, ki je prikazana s

were performed on a PC with a Celeron 600 MHz processor and 384 MB of RAM. The linear behaviour of the algorithm is clearly seen from graphs in Figure 8.

3.2.2 FEM model

For testing the algorithm on real FEM model results it is not necessary to have a large and complex problem. We will clearly show the benefits of the algorithm on a simple example, with more complex problems that have considerably more data, the effectiveness is much higher. The simple example uses a structural FEM analysis of a straight forward part, a steel plate with a centred hole. The applied boundary conditions (continuous load on the upper face and fixed nodes at the bottom) and the mesh of the linear triangular elements are presented in Figure 9. In the mesh, 30,458 linear triangular elements are determined with 15,559 nodes.

The obtained results are presented in Figure 10 with the plotting of the equivalent stress



Sl. 9. *Model MKE z definirano trikotniško mrežo ter robnimi pogoji*
Fig. 9. *FEM model with defined mesh and boundary conditions*

Sl. 10. *Model MKE s Misesovo porazdelitvijo napetosti*
Fig. 10. *FEM model equivalent stress distribution plot*

sivo senčenimi področji zaradi črno bele predstavitve. Za pripravo modela MKE in za njegovo obdelavo je bil uporabljen I-DEAS Master serije 7. Dobljeni so bili pričakovani rezultati, ki prikazujejo simetrično zgostitev napetosti na obeh straneh luknje.

Rezultati so bili izvoženi v treh datotekah ASCII, ki vsebujejo geometrijsko obliko s topologijo, vrednosti deformacij (trije pomiki in trije zasuki) in napetosti (največje in najmanjše glavne napetosti in primerjalne napetosti) v vsakem vozlišču. Datoteke so velikosti 4717 kB, 1136 kB in 1295 kB. Te datoteke smo lahko enostavno spremenili, poenostavili, rekonstruirali in prikazali. Primerjava velikosti datotek je prikazana v preglednici 2. V našem primeru se je pokazalo, da je za prikaz dovolj 25% izvirnih podatkov za sprejemljivo kakovost prikaza rezultatov. Poleg tega je v preglednici 2 prikazana tudi primerjava velikosti datotek s formatom stiskanja GEM^{2}A, ki je rezultat lastnih raziskav [10]. V prispevku so predstavljene le slike z uporabo sivin. Bistveno boljše rezultate lahko vidimo pri uporabi barvne porazdelitve.

V preglednici 2 smo primerjali velikosti datotek med nestisnjeno obliko (ASCII), ki smo jih nato stisnili z obliko PKZIP (ZIP) in z našo obliko (GEM^{2}A). Poleg teh oblik za stiskanje trikotniških mrež, obstajajo še drugi, ki so jih razvili različni avtorji ([11] do [13]). Ti avtorji so se osredotočili predvsem na stiskanje topologije brez geometrijskih podatkov. Izjema je algoritem Toume in Gotsmana [13], ki stiska tudi geometrijske podatke, vendar le-te z izgubami. Zaradi tega primerjave teh algoritmov z našim ni mogoča. Izvedena pa je primerjava učinkovitosti stiskanja same topologije trikotniške mreže med našim algoritmom (GEM^{2}A) in algoritmi, ki so jih razvili De Florianijeva s sodelavci [11], Gumhold in Strasser [12] ter Touma in Gotsman [13]. Primerjavo učinkovitosti prikazuje preglednica 3. Metoda [11] ne omogoča stiskanja trikotniških mrež, ki vsebujejo luknje, zato v preglednici 3 pri teh primerih ni rezultata. Kakor vidimo, predlagana metoda v večini primerov dosega boljša zgostitvena razmerja od drugih metod.

4 SKLEP

Prispevek opisuje algoritem za poenostavljanje trikotniških mrež, ki temelji na odstranjevanju vozlišč. Za pospešitev postopka poenostavitve uporabljamo sekljalno preglednico, s čimer smo dobili linearno časovno zahtevnost celotnega postopka. Predstavili smo tudi hevristiko za vzpostavitev sekljalne preglednice ter njeno polnjenje. Algoritem omogoča tudi postopek obnovitve – postopnega vračanja odstranjenih vozlišč. Uporabnik lahko tako zelo preprosto določi, kakšen odstotek poenostavljanja zadovolji kakovost prikaza za

distribution as a grey shaded area caused by the black and white presentation. The IDEAS Master series 7 was for the FEM's model preparation pre-processing, solving and post-processing. The expected results were obtained, showing symmetrical stress concentrations on both sides of the hole.

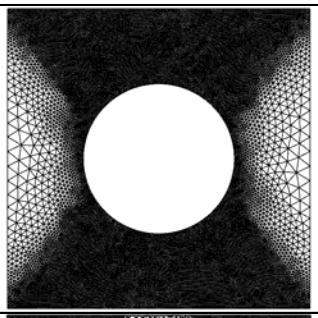
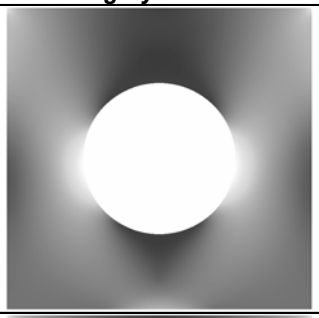
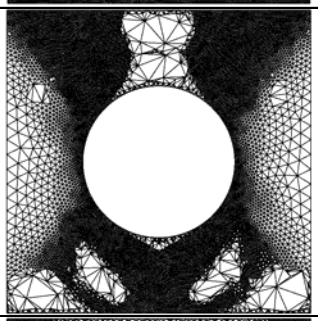
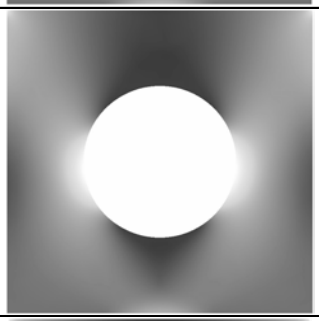
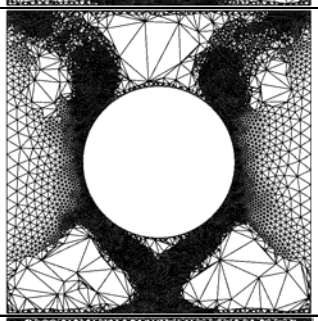
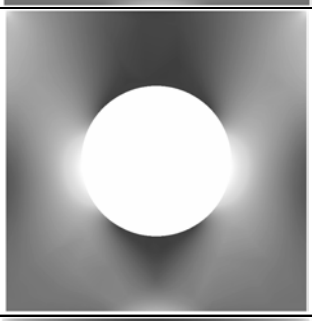
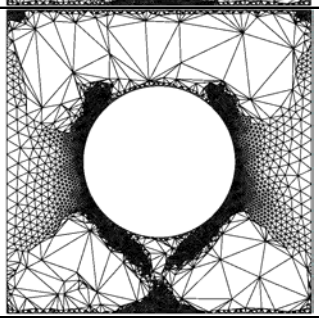
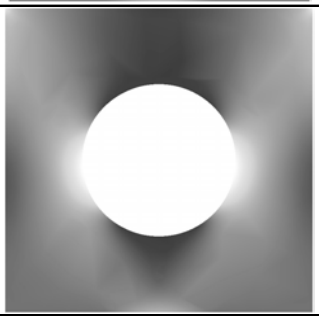
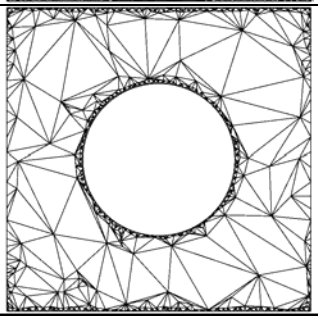
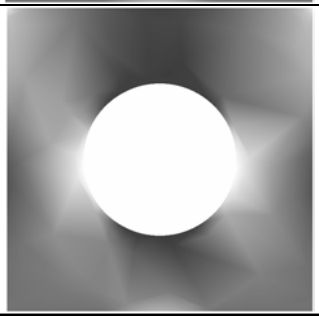
The results data weve exported as three ASCII files consisting of geometry (geometry of nodes and elements topology), deformation values (3 displacements and 3 rotations) and stresses (the major and minor principal stresses and equivalent stress) in all nodes with sizes of 4717kB, 1136kB and 1295kB, respectively. These files were easily converted, decimated, undecimated and viewed. The comparison of the file size is presented in Table 2. In the mentioned example, a ratio of 25% was found to be enough for sharing high-quality results with another user. Moreover, we present in Table 2 the sizes of the GEM^{2}A compressed format, which is a self-developed compression format [10]. It has to be stressed that only greyscale pictures are presented, coloured ones give even better results.

In Table 2, the file sizes using the original ASCII format, the compressed ASCII format (applying the well-known PKZIP (ZIP)) and our compression method (GEM^{2}A) are compared. Clearly, there are also other methods for compressing triangular meshes. The most well known are the methods developed by De Floriani et. al [11], by Gumhold and Strasser [12], and by Touma and Gotsman [13]. However, these authors focused on the compression of the triangular mesh topology, with less attention to geometric data. The exception is the work of Touma and Gotsman [13], which also compresses geometric data, but with losses. For this reason we could not compare these algorithms entirely. Therefore, we made a comparison of the efficiency of the topology compression among the mentioned methods and our approach. The results are summarized in Table 3. The [11] algorithm cannot compress triangular meshes containing holes and therefore not all the examples could be compressed. It is obvious that our method produces the best results in the majority of cases.

4 CONCLUSION

This paper describes an algorithm based on removing nodes, performing a triangular mesh decimation and undecimation. To accelerate the decimation task, a hash table is introduced, and as consequence the selection of the next removed node is done at constant time, and the whole decimation process is solved in linear time. The heuristics for establishing the hash table and its filling are also explained. The heuristics is based on the assumption that engineering applications following the exponential law of evaluation values are treated. The algorithm also supports the process of undecimation,

Preglednica 2. *Primerjava rezultatov, število trikotnikov, prikaz rezultatov in velikosti datotek*
Table 2. *Results of the comparison, the number of triangles, the visualisation and the file size*

	Mreža ***Mesh***	***Prikazani rezultati v sivinah*** ***Visualisation results in greyscale***	
100%			**Vozlišča: Nodes:** 15559 **Trikotniki: Triangles:** 30458 **Velikost: Size:** **ASCII** 1031356 B; **ZIP** 364284 B; **GEM²A** 133435 B
75%			**Vozlišča: Nodes:** 11670 **Trikotniki: Triangles:** 22680 **Velikost: Size:** **ASCII** 754040 B; **ZIP** 273438 B; **GEM²A** 103650 B
50%			**Vozlišča: Nodes:** 7781 **Trikotniki: Triangles:** 14902 **Velikost: Size:** **ASCII** 490614 B; **ZIP** 180307 B; **GEM²A** 71659 B
25%			**Vozlišča: Nodes:** 7781 **Trikotniki: Triangles:** 14902 **Velikost: Size:** **ASCII** 490614 B; **ZIP** 180307 B; **GEM²A** 71659 B
9,4%			**Vozlišča: Nodes:** 3892 **Trikotniki: Triangles:** 7124 **Velikost: Size:** **ASCII** 235076 B; **ZIP** 86716 B; **GEM²A** 36035 B

Preglednica 3. *Primerjava učinkovitosti stiskanja topologije trikotniških mrež*
Table 3. *Comparison of the compression efficiency for the topology compression of triangular meshes*

Ime Name	Vozlišča Nodes	Trikotniki Triangles	Velikost / Size [b]			
			[11]	[12]	[13]	GEM^{2}A
puma / puma	1204	752	/	223	499	306
motor / engine	2536	4566	/	1268	252	252
0100	10000	19602	5002	3734	30	28
maneken2 / mannequin2	11704	23402	6283	5342	667	627
napetost / tension	15559	30458	/	6658	1099	1039
koleno / knee	37888	75264	/	14527	28	26
dinozaver / dinosaur	42146	84288	22794	21240	17380	11067
ozemljilo / grounding	46625	59680	23315	16793	10274	5637
venera / venus	100759	201514	54187	46368	27733	24990

prenos prek spleta. Algoritem se je pri testiranju rezultatov MKE izkazal kot hiter, uporaben in učinkovit. Posebej bi radi poudarili lastnost poenostavljanja, ko lahko inženir hitro na oko oceni najmanjšo potrebno kakovost prikaza rezultatov za prenos drugim uporabnikom prek spleta in s tem bistveno zmanjša količino podatkov, potrebnih za prenos, obenem pa še vedno zagotavlja sprejemljivo kakovost prikaza rezultatov. Algoritem je uporaben tudi pri drugih inženirskih uporabah, ki temeljijo na velikem številu podatkov, na primer omrežen digitalni model reliefa.

i.e., gradual replacement of the nodes being removed. The engineer can easily determine a satisfactory ratio of undecimation for the quality of visualisation necessary to send it over the internet. We tested the algorithm with a few FEM results and found it to be very fast, useful and effective. We should again point out the feature of undecimation, which allows the engineer to easily visually determine the quality level of the results to be sent according to the other internet participant. Moreover, the algorithm can be applied in many engineering applications dealing with triangular meshes with even better results.

5 LITERATURA
5 REFERENCES

[1] Schroeder, W. J., J.A. Zarge, W.E. Lorensen (1992) Decimation of triangle meshes, *Computer Graphics*, 26, 2, 65-70.

[2] Garland, M., P.S. Heckbert (1995) Fast polygonal approximation of terrains and height fields, tehnično poročilo, http://www.cs.cmu.edu/~garland/scape.

[3] Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle (1993) Mesh optimization, *Computer Graphics*, 27, 19-26.

[4] Garland, M., P.S. Heckbert (1997) Surface simplification using quadric error metrics, *SIGGRAPH Conference Proceedings*.

[5] Franc, M., V. Skala (2001) Parallel triangular mesh decimation without sorting, *SCCG Proceedings*, Budmerice, Slovakia, 69-75.

[6] Franc, M., V. Skala (2001) Triangular mesh decimation in parallel environment, *EUROGRAPHICS Workshop on Parallel Graphics and Visualization*, Girona, Spain, 39-52.

[7] Lamot, M., B. Žalik (2000) A contribution to triangulation algorithms for simple polygons. *Journal of Computing and Information Technology – CIT*, 8, 4, 319-331.

[8] ElGindy, H., H. Everett, G. Toussaint (1989) Slicing an ear in linear time, internal memorandum, *School of Computer Science, McGill University*.

[9] de Berg M., van M. Kreveld, M. Overmars, O. Schwarzkopf (1997) Computational geometry: algorithms and applications, *Springer*.

[10] Krivograd, S., B. Žalik (2002) An approach for compression general triangular meshes, patent no. 21203, patent announcement no. P-200200071, Ljubljana: *Slovenian Intellectual Property Office*.

[11] De Floriani, L., P. Magillo, E. Puppo (1998) Compressing TINs, *ACM Symposium on Geographic Information Systems - ACM GIS 98*, Washington, D.C., United States, 145-150.

[12] Gumhold, S., W. Strasser (1998) Real time compression of triangle mesh connectivity, *ACM Computer Graphics - SIGGRAPH'98*, Orlando, Florida, United States, 133-140.

[13] Touma, C., C. Gotsman (1998) Triangle mesh compression, *Graphics Interface'98 - GI'98*, Vancouver, British Columbia, Canada, 26-34.

Naslova avtorjev: dr. Sebastian Krivograd
prof.dr. Borut Žalik
Univerza v Mariboru
Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova 17
2000 Maribor

dr. Gorazd Hren
prof.dr. Anton Jezernik
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17
2000 Maribor

Authors' Addresses: Dr. Sebastian Krivograd
Prof.Dr. Borut □alik
University of Maribor
Faculty of Electrical Eng. and
Computer Science
Smetanova 17
SI-2000 Maribor, Slovenia.

Dr. Gorazd Hren
Prof.Dr. Anton Jezernik
University of Maribor
Faculty of Mechanical Eng.
Smetanova 17
SI-2000 Maribor, Slovenia

Prejeto: 2.9.2002
Received:

Sprejeto: 18.12.2003
Accepted:

Odprto za diskusijo: 1 leto
Open for discussion: 1 year