**Факултет инжењерских наука Универзитета у Крагујевцу**

**Назив студијског програма: Војноиндустријско инжењерство**
**Ниво студија: Мастер академске студије**
**Предмет: Напредна анализа и компјутерска симулација система**
**Број индекса: 317/2016**

**Никола Ј. Палић**

# Оптимизациони процес у идентификацији параметара FEM модела

## Мастер рад

**Комисија за преглед и одбрану:**

1. **Проф. др Мирослав Живковић, ред. проф.**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог мастер рада кандидат треба да изложи преглед основних концепата и метода оптимизације. Поред прегледа најзначајнијих оптимизационих метода, кандидат треба да програмским пакетом за оптимизацију, PEST, оптимизационим процесом изврши идентификацију материјалних параметара FEM модела.

**Препоручена литература:**


[1] Singiresu S. Rao, Engineering Optimization: Theory and Practice, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[2] Којић М., Славковић Р., Живковић М., Грујовић Н.: Метод коначних елемената 1, Машински факултет, Крагујевац, 1998.

[3] Model-Independent Parameter Estimation User Manual Part I: PEST, SENSAN and Global Optimisers, 2016.


Крагујевац, октобар 2017.

Ментор:
Проф. др Мирослав Живковић

_____

# Садржај

## Резиме:

У оквиру овог рада изложен је преглед основних концепата и метода оптимизације. Поред прегледа најзначајнијих оптимизационих метода, презентоване су основне карактеристике једног од најпримењиванијих програмских пакета за оптимизацију, PEST-а. Такође, презентоване су основне теоријске поставке метода коначних елемената.

Детаљно је изложен начин инсталације програмског пакета PEST на персонални рачунар, као и припрема потребних улазних и излазних датотека FEM модела на ком ће се оптимизационим процесом извршити идентификација материјалних параметара. Детаљно је описана методологија креирања свих потребних датотека за покретање прорачуна. На крају су приказане вредности материјалних параметара добијених прорачуном у програмском пакету PEST.


**Кључне речи:** оптимизација, метод коначних елемената, PEST, FEMAP, PAK


## Summary:

An overview of the basic concepts and methods of optimization has been presented within this work. Other than the overview of the most important optimization methods, basic features of one of the most applicable program packages for optimizing PEST have been presented. In addition, finite elements basic theoretical framework has been presented.

The way of installing program package PEST on personal computers is described in detail, as is preparation of necessary input and output files of FEM models on which the optimized process of identification of material parameters will be performed. Metodology of creating all necessary files for beginning with the calculations is described in details. In the end there are values of material parameters produced by calculations in program package PEST.


**Keywords:** optimization, finite elements method, PEST, FEMAP, PAK

# 1. Увод

Оптимизација представља поступак добијања најбољих резултата под одређеним околностима. Идеја оптимизације је да се жељени параметри одређеног система сведу на дозвољени минимум или мксимум, и да се при томе добију најбољи могући резултати. Једна од најбитнијих ствари код оптимизације је да применом исте не нарушимо неке параметре система на које се оптимизација није односила.

У даљем раду најпре ће бити изложен појам оптимизације, дефиниција оптимизационог проблема, као и класификација оптимизационих проблема. Након тога дат је преглед класичних оптимизационих техника.

Затим је објашњена основна функција и намена програмског пакета PEST, који важи за један од најприменљивијих софтвера у области оптимизације.

Након тога изложене су основне карактеристике и могућности програма за креирање модела и постпроцесирање у области анализе методом коначних елемената, FEMAP-а.

Даље, објашњене су основне теоријске поставке методе коначних елемената, и објашњени појмови основног и дегенерисаног 3D елемента. Дат је и кратак преглед солвера за прорачун методом коначних елемената, PAK-а.

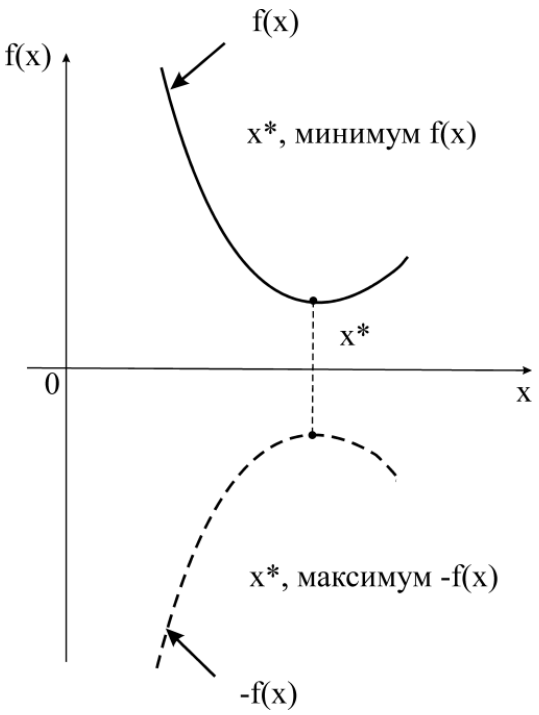
У завршном делу рада, на примерима је оптимизационим процесом извршена идентификација параметара програмским пакетом PEST. Први пример на ком је вршена идентификација параметара је једноставан FORTRAN-ски програм који рачуна вредности експоненцијалне функције.

Наредни пример је модел на ком се врши анализа коначних елемената PAK-ом. Потребно је наћи начин да се на једном оваквом софтверу изврши оптимизација PEST-ом, и да идентификовани параметри задовољавају критеријуме.

# 2. Теоријске поставке оптимизационих алгоритама

Захтеви које пред савремене инжењере поставља тренд смањења трошкова производње, повећања ефикасности производних процеса, модуларности производа и сл., а све у циљу постизања конкурентности, довели су до развоја робусних метода за доношење одлука као што су оптимизационе методе. Ове методе чине основу подршке процесу одлучивања током развоја производа и методологија управљања у циљу истовременог постизања економичности и ефикасност. Оптимизационе методе су достигле одређен степен зрелости последњих деценија који им је омогућио примену у широком спектру људских активности почевши од индустријске производње, преко организационих наука, медицине, па све до проблема везаних за социолошке феномене и еколошке системе. Паралелним напретком компјутерски подржаног развоја, дизајна, моделирања и симулације, оптимизационе методе добијају на посебном значају заузимајући водеће место у пружању подршке креативном концептуалном и примењеном дизајну [1].

Оптимизација представља поступак добијања најбољих резултата под одређеним околностима. Током дизајна, конструисања и експлоатације система, потребно је донети велики број одлука на различитим нивоима имплементације. Основни циљ свих одлука је или минимизација улагања или максимизација добити. Пошто се и улагања и добит за конкретне примере могу описати одређеном функцијом која зависи од променљивих битних за процес одлучивања, оптимизација се може посматрати као процес проналажења решења која максимизирају или минимизирају дату функцију.




*Слика 2.1. Интерпретација минимума* $f(x)$ *као максимума* $-f(x)$ *[2]*

Као што се може и видети са слике 2.1. поступак максимизирања функције $-f(x)$ поклапа са минимизацијом $f(x)$, јер се решење налази у истој тачки $x^*$. Стога се, без губитка општости, оптимизација може третирати као минимизација циљне функције. Не постоји општи поступак за решавање произвољног оптимизационог проблема, па је временом развијен већи број метода у складу са природом третираног оптимизационог проблема.

Методе проналажења оптимума такође се називају и техником математичког програмирања које се проучава као саставни део операционих истраживања. Операциона истраживања представљају математичку дисциплину која се бави применом научних метода у области доношења одлука у циљу проналажења оптималних решења.

Технике математичког програмирања користе се за проналажење минимума функције зависне од већег броја променљивих уз задовољавање дефинисаног скупа ограничења. Стохастичке методе налазе примену у анализи случајних процеса са познатим вероватноћама појављивања. Статистичке методе омогућавају коришћење експерименталних резултата у креирању емпиријских модела који најпрецизније описују физички систем [2].

## 2.1. Историјски развој

Оптимизација налази свакодневну примену у покушајима да се са што мање ангажовања постојећих ресурса постигне што је могуће већи ефекат у процесима планирања производње, алоцирања ресурса, временског планирања, управљања системима, одлучивања и сл. Сам настанак оптимизационих метода може се повезати са радом *Newton*-а, *Lagrange*-а и *Cauchy*-ја. Развој првих оптимизационих метода био је могућ услед рада *Newton*-а и *Leibnitz*-а у области диференцијалног рачуна. Основе минимизације функција постављене су од стране *Bernoulli*-ја, *Euler*-а, *Lagrange*-а и *Weierstrass*-а. Методе оптимизације са постојањем ограничења назване су по свом творцу као *Lagrange*-ови множиоци. *Cauchy* је начинио прве кораке у примени градијентних метода за решавање проблема без постојања ограничења.

Без обзира на ране резултате у области оптимизације, доста дуго није било значајнијег напретка на пољу примене постојећих метода. Највећи број оптимизационих метода за решавање проблема великих димензија заснован је на резултатима постигнутим у време Другог светског рата током организовања логистичке подршке за вишемилионске јединице распоређене на огромним просторима. Свака метода која би довела до унапређења ефикасности у условима ограничених ресурса и кратких временских рокова била је одлучујућа за даљи ток догађаја.

Основе прве примењене оптимизационе методе за решавање проблема великих димензија, такозвана *simplex* метода, развијене су још током рата. Пуну примену ова метода налази непосредно по завршетку рата услед развоја рачунарске технике, тачније 1947. године као резултат рада *Dantzig*-а. О ускоj повезаности рачунара и оптимизације

говори и чињеница да су у првим годинама свог развоја рачунари скоро у потпуности коришћени за прорачуне везане за проблеме решаване *simplex* методом.

Примењена оптимизација и даље представља област отворену за иновативне доприносе, јер се проблеми и начини решавања могу веома разликовати од случаја до случаја, па се свакодневно јављају помаци у виду нових и модификације постојећих оптимизационих метода и алгоритама. Развој метода унутрашње тачке у проблемима линеарног програмирања, који је развијен касних ’80-их, представља један од таквих спектакуларних помака на пољу примењене оптимизације. Такође, један од најновијих трендова је настао као интеракција између математичке теорије оптимизације и новог поља у софтверском инжењерингу, такозваног програмирања са ограничењима (*constraint programming*), које се показало као одличан алат у решавању проблема оптимизације. Неке од највећих софтверских фирми које производе оптимизационе пакете намењене одлучивању у широком спектру инжењерске и менаџерске примене, управо су засновале своје производе на овом концепту.

Симулирано очвршћавање, генетички алгоритми и неуронске мреже представљају групу нових метода математичког програмирања које су током последње деценије добиле на значају услед своје општости и једноставне имплементације. Симулирано очвршћавање се базира на опонашању процеса очвршћавања у материјалу. Генетички алгоритми имплементирају еволуционе феномене наслеђивања и селекције, док неуронске мреже решавају проблеме балансирањем веза у сложеним мрежама неурона, по угледу на поједностављени начин функционисања нервног система [1].

## 2.2. Дефиниција оптимизационог проблема

Оптимизациони проблем, или проблем математичког програмирања може се дефинисати као:

$$\text{Пронаћи } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \text{ који минимизира } f\left(\mathbf{X}\right) \tag{2.1}$$

и при томе задовољава ограничења

$$\begin{aligned} g_j\left(\mathbf{X}\right) &\le 0, \quad j = 1, 2, ..., m \\ l_k\left(\mathbf{X}\right) &= 0, \quad k = 1, 2, ..., p \end{aligned} \tag{2.2}$$

где је $\mathbf{X}$ *n*-димензионални вектор променљивих, $f\left(\mathbf{X}\right)$ се назива циљном функцијом а $g_j\left(\mathbf{X}\right)$ и $l_k\left(\mathbf{X}\right)$ као услови неједнакости и једнакости респективно. Број променљивих *н* и број ограничења *m* и *n* у општем случају нису ни у каквој вези. Проблем који је дефинисан са (2.1) и (2.2) назива се оптимизациони проблем са ограничењима. Одређена група проблема не узима у обзир ограничења, па се може дефинисати као

$$\text{Пронаћи } \mathbf{X}=\begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \text{који минимизира } f\left(\mathbf{X}\right) \tag{2.1}$$

и самим тим се називају оптимизациони проблеми без ограничења [2].

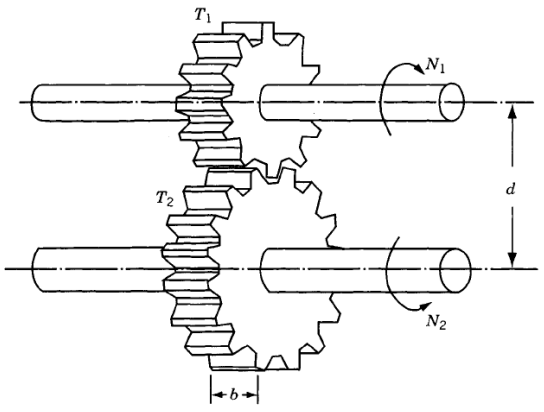
### 2.2.1. Вектор управљачких променљивих

Параметри који се користе за опис физичког система могу се третирати као непроменљиви или предефинисани или могу бити променљиви и као такви представљати предмет оптимизационог процеса. Све променљиве које се могу мењати током процеса оптимизације $x_i, i=1,2,\ldots,n$ називају се управљачке променљиве. Ове променљиве формирају вектор управљачких променљивих $\mathbf{X}$.

На слици 2.2. је као пример приказан проблем два спрегнута зупчаника који су описани помоћу ширине зупчаника *b*, броја зубаца $T_1$ и $T_2$, осног растојања *d*, облика профила зупца и типа материјала. Ако се осно растојање, облик профила и тип материјала фиксирају као предефинисани параметри, остали параметри чине вектор променљивих

$$\mathbf{X}=\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}=\begin{Bmatrix} b \\ T_1 \\ T_2 \end{Bmatrix} \tag{2.4}$$

Ако нема дефинисаних ограничења везано за вредности које могу узимати ове променљиве, свака њихова комбинација може се сматрати могућим решењем. У општем случају када имамо *n* променљивих $x_i, i=1,2,\ldots,n$, вишедимензионални (тачније *n*-димензионални) простор који оне дефинишу назива се простором претраживања. Свака тачка у овом простору представља могуће, или у случају постојања ограничења, немогуће решење разматраног проблема.




*Слика 2.1. Проблем спрегнутих зупчаника [2]*

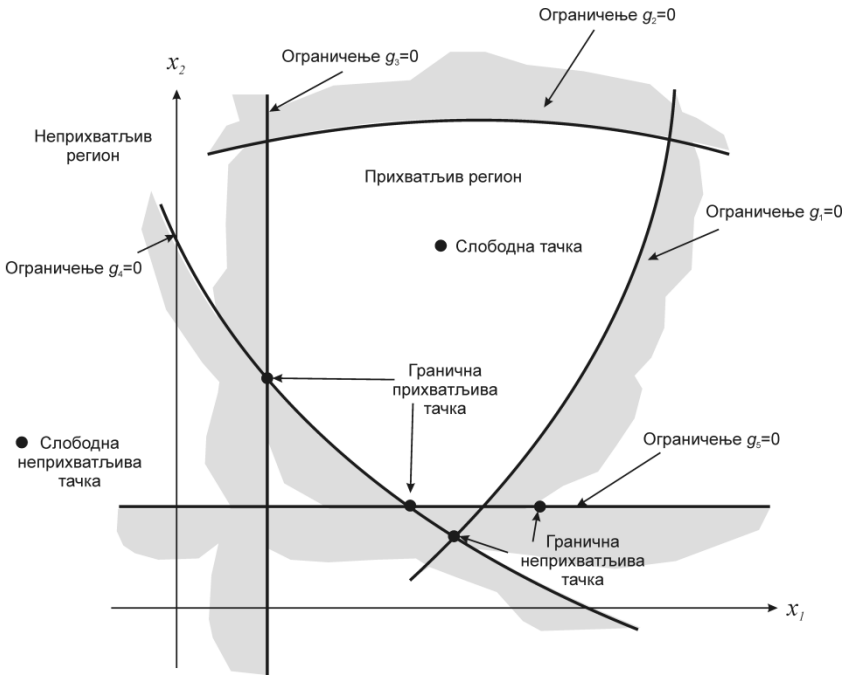
У конкретном случају, тачка $(1,0\ 20\ 40)$ представља могуће решење, док тачка $(1,0-20\ 40,5)$ представља немогуће решење проблема, јер је неизводљиво дефинисати зупчаник са негативним или децималним бројем зубаца [2].

### 2.2.2. Ограничења

У већини практичних проблема, управљачке променљиве не могу узимати произвољне вредности. Рестрикције које важе за одређене управљачке променљиве називају се ограничењима. Ограничења која се односе на лимитирања у смислу функционалности и понашања система називају се функционалним ограничењима. Лимити који се односе на физичку природу система називају се геометријским или споредним ограничењима. У случају приказаног пара зупчаника, ширина зупчаника *b* мора имати минималну могућу вредност у светлу максималних напона који се могу јавити, а који зависе и од максималног прописаног момента. Такође, однос броја зубаца ($T_1/T_2$) повезан је са преносним фактором који се очекује од система ($N_1/N_2$). Ова ограничења се сматрају функционалним ограничењима. Са друге стране, параметри $T_1$ и $T_2$ могу узимати само целобројне вредности, па чак и њихове екстремне вредности могу зависити од могућности технолошког процеса. Пошто се наведена ограничења односе на физичке карактеристике система она спадају у споредна ограничења [2].

### 2.2.3. Ограничавајуће површи

Као први пример могуће је посматрати оптимизациони проблем са само једним ограничењем у виду неједнакости $g_j(\mathbf{X}) \le 0$. Скуп вредности $\mathbf{X}$ који задовољава неједнакост $g_j(\mathbf{X}) \le 0$ чини хиперповрш у простору претраживања названу ограничавајућа површ. Ова површ дели простор претраживања на две области: једну у којој важи $g_j(\mathbf{X}) < 0$ и $g_j(\mathbf{X}) > 0$. Стога тачке које чине ову површ представљају екстремна прихватљива решења проблема, док тачке у области $g_j(\mathbf{X}) < 0$ представљају прихватљива решења а тачке у области $g_j(\mathbf{X}) > 0$ представљају немогућа, односно неприхватљива решења.




*Слика 2.3. Ограничавајуће површи у хипотетичком дводимензионалном простору претраживања [2]*

На слици 2.3. приказане су ограничавајуће површи у хипотетичком дводимензионалном простору претраживања, при чему је област неприхватљивих решења приказана као шрафирана. Решења која су представљена тачкама које се налазе на једној или више ограничавајућих површина називају се гранична решења, док се остала решења сматрају слободним. У зависности од области у којој се налазе, решења могу бити:

- Слободна и прихватљива
- Слободна и неприхватљива
- Гранична и прихватљива
- Гранична и неприхватљива

Сваки од ова четири типа илустрован је на слици 2.3.

### 2.2.4. Циљна функција

Обзиром да се при решавању оптимизационих проблема генерално може доћи до већег броја прихватљивих решења, потребно је дефинисати начине избора најоптималнијег решења. Стога се мора дефинисати критеријум којим се могу поредити два могућа решења. Када се овај критеријум дефинише као функција управљачких променљивих може се говорити о циљној или критеријумској функцији. Избор облика функције повезан је са самом природом проблема који се решава. У највећем броју случајева облик функције може бити сложен, као на пример у случају механичког преносника, максималан степен искоришћење није истовремено оптималан у погледу масе преносника, што може бити од значаја за оптималност решења. Због наведеног може се сматрати да је избор облика циљне функције управо најзначајнији корак у решавању оптимизационог проблема.

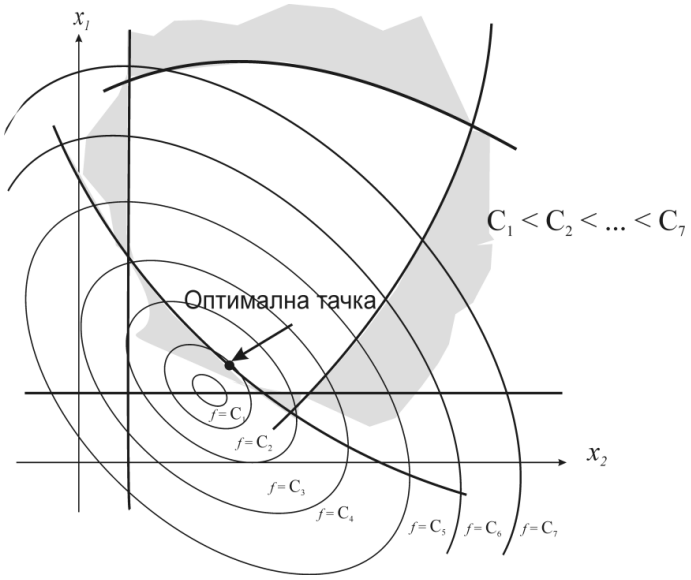
Веома често је потребно истовремено задовољити више критеријума. Рецимо да механички преносник мора дати најбољи степен искоришћења при одређеној снази, а истовремено са минималном масом. Овакви проблеми спадају у област вишекритеријумске оптимизације. Једно од најједноставнијих решења оваквих проблема је да се она преведу у једнокритеријумску оптимизацију формирањем циљне функције као линеарне суме појединачних циљних функција. Тако се може рећи да ако постоје два критеријума $f_1(\mathbf{X})$ и $f_2(\mathbf{X})$, онда се може формирати следећа циљна функција

$$f(\mathbf{X}) = \alpha_1 f_1(\mathbf{X}) + \alpha_2 f_2(\mathbf{X}) \tag{2.5}$$

где су $\alpha_1$ и $\alpha_2$ тежински коефицијенти који указују на значај одређеног парцијалног критеријума [2].

### 2.2.5. Циљне површи

Скуп тачака при којима је циљна функција константна, односно задовољена је једнакост $f(\mathbf{X}) = c = const$, чини хиперповрш у простору претраживања, и могуће је за свако појединачно *x* конструисати дату одговарајућу површ. Ова површи се називају циљним површима и приказане су на слици 2.4. у случају хипотетичког дводимензионалног простора претраживања.




*Слика 2.2. Циљне површи у хипотетичком дводимензионалном простору [2]*

Када се конструишу циљне површи, заједно са ограничавајућим површима, на датом примеру је врло једноставно идентификовати оптимално решење. Међутим, овај пример је симплификован, и у реалним проблемима је број димензија далеко већи, па је немогуће уопште визуелизовати дате површи или говорити о визуелном проналажењу оптимума [2].

## 2.3. Класификација оптимизационих проблема

Оптимизациони проблеми се могу класификовати на више начина, тако да је могуће формирати различите структуре типова проблема. Наредна класификација је само једна од могућих [3].

### 2.3.1. Класификација према постојању ограничења

Као што је већ речено, проблеми могу бити са ограничењима и без ограничења, у зависности од постојања скупа ограничења над управљачким променљивама [2].

### 2.3.2. Класификација према типу управљачких променљивих

Према природи управљачких променљивих оптимизациони проблеми се могу поделити у две категорије. У прву категорију спадају проблеми код којих је потребно пронаћи вредности за скуп управљачких променљивих који минимизира дату циљну

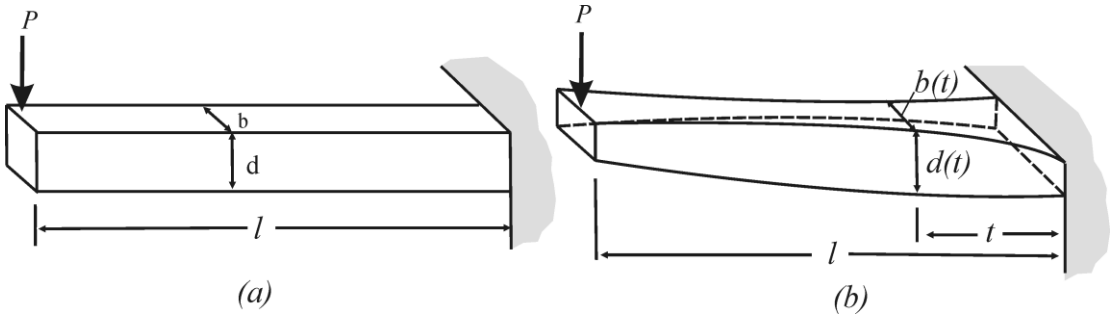
функцију испуњавајући задата ограничења. Као пример може се посматрати минимизација тежине призматичне конзоле као што је приказано на слици 2.5. Проналажење минималне тежине условљено је максималним дозвољеним угибом на крају конзоле.

$$\text{Пронаћи } \mathbf{X}=\begin{Bmatrix} b \\ d \end{Bmatrix} \text{ које минимизира } f\left(\mathbf{X}\right)=\rho lbd \tag{2.6}$$

при чему мора бити задовољено

$$\begin{aligned} &\delta\left(\mathbf{X}\right)\le\delta_{\max} \\ &b\ge 0 \\ &d\ge 0 \end{aligned} \tag{2.7}$$

где $\rho$ представља густину а $\delta$ је угиб на крају конзоле. Овакви проблеми се називају параметарским или статичким оптимизационим проблемима.




*Слика 2.3. Пример оптимизације геометрије конзоле [2]*

У другу групу проблема спадају они који имају за циљ да пронађу оптималне вредности управљачких променљивих које представљају континуалне функције неког другог параметра система, такве да се минимизује циљна функција при поштовању скупа дефинисаних ограничења. Уколико се дозволи да попречни пресек конзоле буде променљив дуж осе конзоле, као на слици 2.5. проблем се може дефинисати као

$$\begin{aligned} &\text{Пронаћи } \mathbf{X}=\begin{Bmatrix} b(t) \\ d(t) \end{Bmatrix} \text{ које минимизира} \\ &f\left[\mathbf{X}(t)\right]=\rho\int_0^l b(t)\,d(t)\,dt \end{aligned} \tag{2.8}$$

при чему мора бити задовољено

$$\begin{aligned} &\delta\left[\mathbf{X}(t)\right]\le\delta_{\max} \quad 0\le t\le l \\ &b(t)\ge 0 \qquad\qquad 0\le t\le l \\ &d(t)\ge 0 \qquad\qquad 0\le t\le l \end{aligned} \tag{2.9}$$

Код ове категорије проблема управљачке променљиве су функције параметра $m$. Овакви проблеми се називају динамички оптимизациони проблеми [2].

### 2.3.3. Класификација према физичкој структури система

Значајну класу задатака оптимизације представљају проблеми оптималног управљања системима. То су задаци одређивања управљања као функције времена. Управљањем треба посматрани систем, уз поштовање одређених услова и ограничења, у неком временском интервалу превести из почетног у жељено стање а да при томе изабрани критеријум, тзв. индекс перформансе, добије екстремну вредност. Решење задатка оптималног управљања нису бројеви већ функције. То је битна разлика у односу на до сада разматране задатке оптимизације. Неком управљању, тј. некој одређеној временској функцији, придружује се број који означава квалитет управљања. Другим речима, индекс перформансе је пресликавање скупа функција на скуп тачака. Проблем оптималног управљања је одређивање функције управљања којом се посматрани систем преводи из почетног у жељено стање на начин који се сматра најбољим у односу на изабрани критеријум. У неким задацима се тражи истовремено одређивање управљања $u(t)$ и параметара система А. Не само у тим случајевима, него уопште, математички модели реалних динамичких система управљања су ретко такви да се овако постављени задатак може решити аналитички. Зато се предузимају различите апроксимације од којих је најчешће замењивање континуалних модела дискретном формом и његово решавање нумеричким методама. Пошто се у овој књизи разматрају скоро искључиво статички проблеми, само ради илустрације овде се наводи поставка класичног "проблема брахистохроне". Ради се о проблему криве најкраћег времена или криве најбржег спуста који је формулисао *Johann Bernoulli* 1696. г. и за који се везује рађање теорије оптималног управљања [2].

### 2.3.4. Класификација према типу једначина које описују систем

Још један битан начин класификације врши се према типу једначина којима се дефинишу циљна функција и ограничења. Према овој подели, оптимизациони проблеми могу бити линеарни, нелинеарни, геометријски и квадратни. Ова класификација је веома битна са тачке гледишта имплементације алгоритама на рачунару, јер је за наведене области развијен велики број специјализованих рутина. Одређивање типа проблема према наведеној класификацији углавном директно води ка избору процедура за практичну имплементацију [2].

### 2.3.5. Класификација према типу могућих управљачких променљивих

У задацима линеарног програмирања може да се постави и додатни захтев по коме неке или све променљиве морају да узимају вредности из скупа целих или из скупа реалних бројева. Ако је управљачким променљивама дозвољено да узимају вредности само из скупа целих бројева, говори се о целобројном програмирању. Са друге стране, ако вредности могу бити и реални бројеви, говори се о програмирању са реалним вредностима [2].

### 2.3.6. Класификација на основу одређености променљивих

У зависности од тога да ли су параметри у оптимизационом проблему познати, случајног карактера или неодређени, разликују се детерминистички задаци, задаци стохастичке оптимизације и задаци оптимизације у условима неодређености.

У даљем тексту ћемо се бавити претежно детерминистичким задацима који одговарају системима и проблемима у којима су параметри и процеси детерминистичке природе.

Међутим, у стохастичким системима параметри система или посматрани процес имају случајан карактер описан методама из теорије вероватноће и статистике. Постоје две велике групе метода стохастичке оптимизације: имплицитне и експлицитне методе. Имплицитне методе се могу примењивати само за дискретне стохастичке проблеме у којима је број могућих решења релативно мали, тј. такав да се свако решење може анализирати. У примени ових метода разликују се следећа три основна корака:

- Одреде се све могуће реализације случајних величина;
-Ураде се детерминистичке оптимизације за сваку реализацију;
- Анализирају се резултати и изабере решење.

Типична примена ових метода је на проблеме одлучивања у условима неизвесности и ризика који су моделирани стаблом одлучивања.

Експлицитне методе стохастичког програмирања се примењују за дискретне и континуалне стохастичке проблеме. Притом се може десити да стохастичку природу има критеријумска функција и/или функције ограничења.

### 2.3.7. Класификација према броју циљних функција

У зависности од броја циљних функција које се желе минимизирати, оптимизациони проблеми могу бити подељени у две групе: једнокритеријумске и вишекритеријумске оптимизационе проблеме. До сада разматрани проблеми могу се сврстати у једнокритеријумске проблеме.

Вишекритеријумски оптимизациони проблем може се дефинисати као

$$\begin{aligned} &\text{Пронаћи} \quad \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \quad \text{које минимизира} \quad f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots f_n(\mathbf{X}) \\ &\text{уз ограничења} \quad \mathrm{g}_j(\mathbf{X}) \le 0, \quad j = 1, 2, \ldots, m \end{aligned} \tag{2.10}$$

где су $f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots f_n(\mathbf{X})$ циљне функције које се симултано минимизују [2].

## 2.4. Класичне оптимизационе технике

Класичне оптимизационе технике се примењују у случајевима када се као предмет решавања јављају континуалне и диференцијабилне функције. Ове технике су аналитичке и користе методе диференцијалног рачуна у циљу проналажења оптималног решења. Пошто већину проблема није могуће описати оваквим функцијама, класичне оптимизационе технике немају велики значај у пракси. Међутим, неопходно је упознати се са класичним техникама као основом за разумевање и имплементацију других напреднијих техника. Класични приступ решавању оптимизационих проблема обухвата следеће основне кораке:

- наћи потребне услове које мора да задовољи оптимално решење користећи диференцијалне особине функције у оптималном решењу;
- решити једначине које чине потребан услов да би се добили кандидати за оптимално решење;
- тестирати проверити кандидате за оптимално решење коришћењем тестова за потребне и довољне услове.

Процедуре које се развијају на основу овог приступа називају се индиректне методе јер се њима оптимално решење проналази индиректно на основу диференцијалних особина функције циља или функционала, а не директно на основу вредности функције. У директним методама се непосредно користе вредности критеријума и ограничења датог проблема да би се систематским рекурзивним поступцима дошло до оптималног решења. Није увек могуће нити је неопходно да се

направи јасна разлика између ових метода; у многим оптимизационим процедурама комбинују се оба приступа.

Класичне оптимизационе технике се могу применити у случају функције са једном променљивом, вишепараметарским функцијама без ограничења и вишепараметарским функцијама са ограничењима у виду једнакости и неједнакости [2].

### 2.4.1. Оптимизација функције са једном променљивом

Ако се посматра функција са једном променљивом $f(x)$ онда се може говорити о различитим типовима екстремума (минимума или максимума). Тако се може рећи да је тачка $x^*$ је глобални минимум ако је

$$f(x^*) \leq f(x) \tag{2.11}$$

за све *x*.

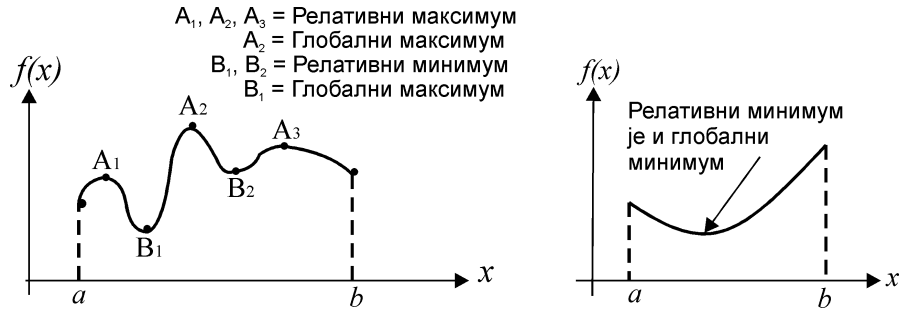
Тачка $x^*$ је локални (слаби) минимум ако постоји околина $N(x^*)$ тако да је

$$f(x^*) \leq f(x), \quad x \in N(x^*) \tag{2.12}$$

Тачка $x^*$ је строги локални минимум ако постоји околина $N(x^*)$ тако да је

$$f(x^*) < f(x), \quad x \in N(x^*), x \neq x^* \tag{2.13}$$

Тачка $x^*$ је изоловани локални минимум ако постоји околина $N(x^*)$ тако да је $x^*$ једини локални минимум у *N*.




*Слика 2.6. Релативни и глобални минимум и максимум [2]*

Најчешћи случај код практичних примера је да постоји више локалних минимума, од којих је само један глобални. Специјалан случај су конвексне функције, где је сваки локални минимум уједно и глобални минимум.

Проблем оптимизације једнопараметарске функције састоји се у проналажењу оптималне вредности $x = x^*$ унутар интервала $[a,b]$ при којој је $f(x)$ минимизирана. Потребни и довољни услов постојања минимума унутар задатог интервала засновани су на примени *Taylor*-овог реда.

Потребни услови за оптималност решења када је функција диференцијабилна разматрани су још у елементарној математици. Нека је $f(x)$ непрекидна функција чији је и први извод такође непрекидна функција (за такве функције се каже да припадају класи $C^1$, док за непрекидне функције чији је *n*-ти извод такође непрекидна функција каже се да припадају класи $C^n$). Ако се претпостави да се максимум (минимум) функције $f(x)$ не постиже за $x = \pm\infty$, онда се максимум (минимум) може појавити само у тачкама у којима је нагиб $\frac{df(x)}{dx}$ једнак нули. Према томе, потребан услов за оптимум је $\frac{df(x)}{dx} = 0$ за оптималну вредност *x*.

Довољан услов за оптималност је онај чије задовољење гарантује да је посматрано решење оптимално; међутим, ако посматрано решење не задовољава довољан услов за оптималност, то не значи да оно мора да буде неоптимално. Само ако решење задовољава услов оптималности који је истовремено и потребан и довољан може се тврдити да је оптимално, и обрнуто, само оптимална решења задовољавају потребне и довољне услове за оптималност.

Када је у тачки $x^*$ задовољен потребни услов, тј. први извод у тој тачки једнак је нули, довољан услов се добија на основу чињенице да други извод одређује знак промене у околини те тачке. То значи да је довољан услов за максимум да је други извод негативан, тј, промена $x^*$ имала би за последицу смањење функције; обрнуто, довољан услов за минимум је да је други извод позитиван [2].

### 2.4.2. Оптимизација вишепараметарских функција без ограничења

Аналогно претходном разматрању може се приступити проблему када се решава задатак безусловне оптимизације функције $f(\mathbf{X}) = f(x_1, x_2, \ldots, x_n)$ која је двоструко диференцијабилна. *Taylor*-ов развој функције $f(\mathbf{X})$ у тачки $\mathbf{X} = \mathbf{X}^* + \mathbf{h}$, где је $\mathbf{X}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ и $\mathbf{h} = (h_1, h_2, \ldots, h_n)$ може се представити као

$$f(\mathbf{X}^* + \mathbf{h}) = f(\mathbf{X}^*) + \sum_{j=1}^{n} h_j \frac{\partial f}{\partial x_j} + \sum_{j=1}^{n}\sum_{k=1} h_j h_k \frac{\partial^2 f}{\partial x_j \partial x_k} + \ldots \qquad (2.14)$$

односно

$$f\left(\mathbf{X}^{*}+\mathbf{h}\right)-f\left(\mathbf{X}^{*}\right)=\mathbf{h}^{\mathrm{T}}\nabla f\left(\mathbf{X}^{*}\right)+\frac{1}{2}\mathbf{h}^{\mathrm{T}}\mathbf{G}\left(\mathbf{X}^{*}\right)\mathbf{h}+o^{3}\left(\mathbf{h}\right) \tag{2.15}$$

где је

$\nabla f\left(\mathbf{X}\right)$ - градијент функције $f\left(\mathbf{X}\right)$, вектор димензије 1×*н*,

$\mathbf{G}\left(\mathbf{X}\right)$ - *Hesse*-ова матрица или Хесијан, симетрична матрица димензије *н*×*н* са елементима $G_{jk}=\dfrac{\partial^{2} f}{\partial x_{j}\partial x_{k}}$.

Из дефиниције следи да функција у екстремној тачки не треба да расте нити опада. Да би $\mathbf{X}^{*}$ било екстремна тачка, разлика $f\left(\mathbf{X}^{*}+\mathbf{h}\right)-f\left(\mathbf{X}^{*}\right)$ треба да је за мало $\boldsymbol{x}$ приближно једнака нули. То значи треба да је

$$\nabla f\left(\mathbf{X}^{*}\right)=0 \tag{2.16}$$

Према томе, потребан услов да је $\mathbf{X}^{*}$ екстремна тачка је $\nabla f\left(\mathbf{X}^{*}\right)=0$.

Довољан услов се добија из захтева да у сваком смеру око $\mathbf{X}^{*}$ функција опада ако се тражи максимум, односно расте ако се тражи минимум. Пошто је $\nabla f\left(\mathbf{X}^{*}\right)=0$, знак промене је одређен вредношћу $\frac{1}{2}\mathbf{h}^{\mathrm{T}}\mathbf{G}\left(\mathbf{X}^{*}\right)\mathbf{h}$, па је довољан услов за максимум да је ова вредност негативна

$$\frac{1}{2}\mathbf{h}^{\mathrm{T}}\mathbf{G}\left(\mathbf{X}^{*}\right)\mathbf{h}<0 \tag{2.17}$$

односно, да је $\mathbf{G}\left(\mathbf{X}^{*}\right)$ негативно дефинитна форма. Када се тражи минимум функције, матрица $\mathbf{G}\left(\mathbf{X}^{*}\right)$ треба да је позитивно дефинитна.

Оба наведена услова се односе на локалне оптимуме. Да би они важили и за глобалне оптимуме, потребно је да критеријумска функција буде конкавна (када се тражи максимум) или конвексна (за задатак минимизације) јер тада на посматраном скупу постоји само једна екстремна тачка која задовољава дате услове [4].

### 2.4.3. Оптимизација вишепараметарских функција са ограничењима у облику једнакости

Оптимизација вишепараметарских функција са ограничењима у облику једнакости може се представити као

$$\begin{aligned} &\text{Минимизирати } f = f(\mathbf{X}) \\ &\text{тако да је испуњено } g_j(\mathbf{X})=0, \quad j=1,2,\ldots,m \end{aligned} \tag{2.18}$$

где је

$$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}$$

При претходно наведеном, *m* је мање или једнако *n*. У супротном, ако је $m > n$ у општем случају није могуће пронаћи решење. Постоји више приступа решавању оваквих проблема, од којих су најпознатији директне замене, ограничених варијација и *Lagrange*-ових множитеља [4].

### 2.4.4. Оптимизација вишепараметарских функција са ограничењима у облику неједнакости

Проблем оптимизације вишепараметарских функција са ограничењима у облику неједнакости дефинисан је као

$$\begin{aligned} &\text{Минимизирати } f(\mathbf{X}) \\ &g_j(\mathbf{X}) \le 0, \quad j=1,2,\ldots,m \end{aligned} \tag{2.19}$$

Неједнакости које се јављају као ограничења могу се превести у једнакости увођењем, за сада непознатих, ненегативних функција $y_j^2$

$$g_j(\mathbf{X}) + y_j^2 = 0, \quad j=1,2,\ldots,m \tag{2.20}$$

На овај начин се проблем преводи у следећи облик

$$\begin{aligned} &\text{Минимизирати } f(\mathbf{X}) \\ &G_j(\mathbf{X},\mathbf{Y}) = g_j(\mathbf{X}) + y_j^2 = 0, \quad j=1,2,\ldots,m \end{aligned} \tag{2.20}$$

Дати проблем се поуздано решава са методом *Lagrange*-ових множитеља, па се стога прво дефинише функција *L*

$$L(\mathbf{X},\mathbf{Y},\boldsymbol{\lambda}) = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j G_j(\mathbf{X},\mathbf{Y}) \tag{2.21}$$

где $\boldsymbol{\lambda} = \begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{Bmatrix}$ представља вектор *Lagrange*-ових множитеља. Стационарна тачка *Lagrange*-ове функције може се пронаћи решавањем система једначина (потребан услов).

$$\frac{\partial L}{\partial x_i}(\mathbf{X}, \mathbf{Y}, \boldsymbol{\lambda}) = \frac{\partial f}{\partial x_i}(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i}(\mathbf{X}) = 0, \quad i = 1, 2, \ldots, n \tag{2.22}$$

$$\frac{\partial L}{\partial \lambda_j}(\mathbf{X}, \mathbf{Y}, \boldsymbol{\lambda}) = G_j(\mathbf{X}, \mathbf{Y}) = g_j(\mathbf{X}) + y_j^2 = 0, \quad j = 1, 2, \ldots, m \tag{2.23}$$

$$\frac{\partial L}{\partial y_j}(\mathbf{X}, \mathbf{Y}, \boldsymbol{\lambda}) = 2\lambda_j y_j = 0, \quad j = 1, 2, \ldots, m \tag{2.44}$$

Може се видети да претходна три израза представљају систем од $(n+2m)$ једначина са исто толико непознатих. Као решење система добијају се оптимални вектор управљачких променљивих $\mathbf{X}^*$, вектор *Lagrange*-ових множитеља $\boldsymbol{\lambda}^*$ и вектор допунских променљивих $\mathbf{Y}^*$.

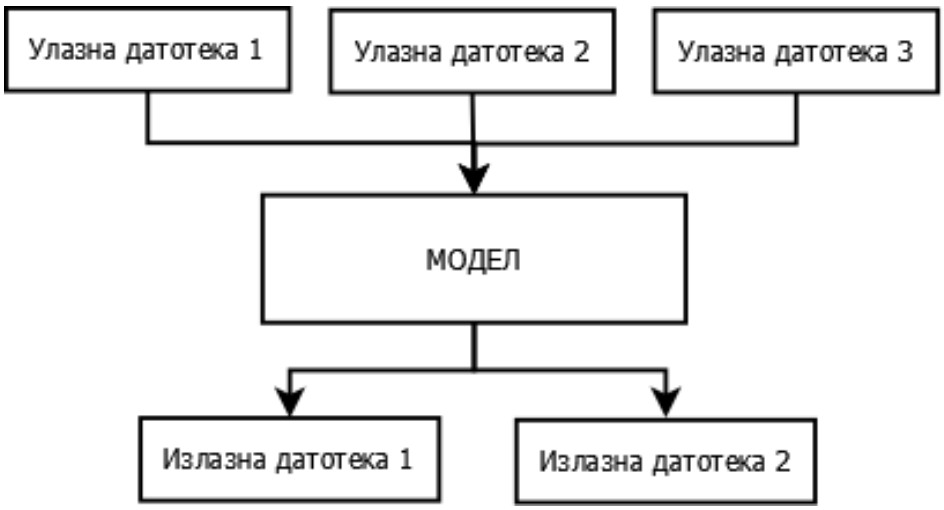
Једначина (2.43) обезбеђује задовољавање ограничења, док једначина (2.44) указује да је или $\lambda_j = 0$ или $y_j = 0$. Уколико је $\lambda_j = 0$ то значи да је *j*-то ограничење неактивно у датој тачки, па самим тим може бити игнорисано, док је случају $y_j = 0$ ограничење у оптимуму активно $(g_j = 0)$ [4].

# 3. PEST – програски пакет за оптимизацију

PEST је нелинеарни параметарски пакет процене. Погодност коришћења PEST-а је у томе што се може користити за процену параметара за било који постојећи рачунарски модел, без обзира да ли корисник има приступ изворном коду модела. PEST је у стању да "преузме контролу" над моделом, покреће га онолико пута колико је потребно док прилагоди своје параметре, све док се неусаглашености између изабраних излаза модела и комплементарног скупа поља или лабораторијских мерења сведе на минимум.

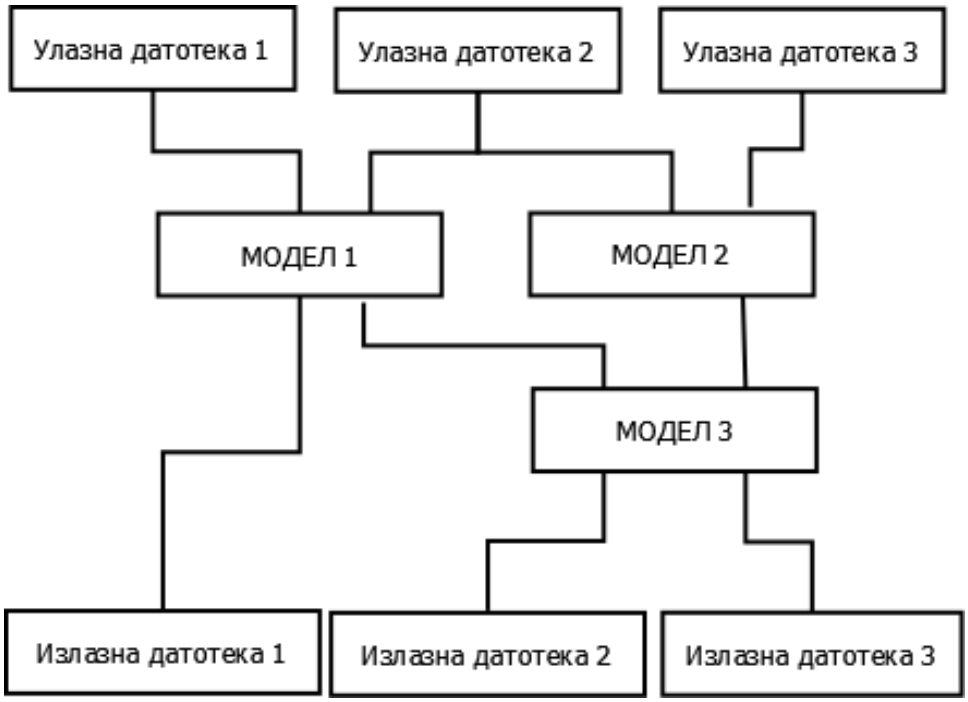
Већина пакета за процену параметара пати од два озбиљна недостатака који онемогућавају њихову способност оптимизације параметара. Прва од ових потешкоћа је да модел обично треба делимично кориговати како би комуницирали са програмом процене; ово обично подразумева преобликовање модела као подпрограма који затим процењивач позива сваки пут када треба да покрене модел. Други недостатак је чињеница да се перформансе многих програма процене озбиљно деградирају приликом оптимизације параметара за велике нумеричке моделе или за сложене моделе који се користе за симулирање еколошких процеса.

PEST превазилази прву од ових потешкоћа комуницирањем са моделом кроз сопствене улазне и излазне датотеке модела. PEST се прилагођава моделу, модел не мора бити прилагођен PEST-у. PEST решава други проблем применом *Gauss-Marquardt-Levenberg* методе нелинеарне процене параметара. Поред тога, прилагођавањем бројних управљачких варијабли, корисник може "прилагодити" PEST-ову примену методе према моделу за који се параметри траже [5].




*Слика 3.1. PEST комуницира са постојећим моделом кроз сопствене улазне и излазне датотеке модела [5]*

Због тога што је PEST ради независно од модела, "модел" у ствари може бити серија модела који PEST започиње у сукцесији кроз пакетну датотеку; PEST може проценити параметре за један или све моделе истовремено. Према томе, први модел може обезбедити улазне податке за други модел. Један модел може бити калибрисан у односу на различите скупове података одједном што кориснику даје огромне могућности [5].




*Слика 3.2. PEST може да калибрише моделе затворене у серијским или скриптним датотекама произвољне сложености [5]*

## 3.1. Карактеристике PEST-а

**PEST - нелинеарни параметарски пакет процене поседује следеће карактеристике:**

- Појединачни параметри модела могу бити означени као подесиви, фиксни или повезани са другим параметрима; подесиви параметри могу бити трансформисани да би повећали ефикасност оптимизације.
- Претходне информације о параметрима или о односима између параметара могу се укључити у процес процене.
- Оптималне вредности параметара могу бити ограничене да леже између појединачно одређене горње и доње границе; ово се примјењује помоћу математички напредног алгоритма који уствари регулише проблем процене параметара пошто су границе одређене.
- Прорачун PEST-а може се прекинути у било ком тренутку како би се извршио детаљан преглед досијеа; PEST се затим може поново покренути тамо где је прекинут.
- Сва складиштења података PEST-а су динамички додељена; стога је величина проблема (укључујући број подесивих параметара и величину скупа података о опсервацији) ограничена само количином меморије инсталиране на корисничкој машини.
- Корисник може да интервенише у процесу процене параметара, држећи неповратне параметре одређене неко вријеме; делови процеса инверзије се затим могу поновити коришћењем претходно израчунатих информација о осетљивости.
- Параметри и опсервације могу се поделити на подгрупе за расподелу варијабли које контролишу израчунавање изведених деривата с једне стране и за процену доприноса различитих опсервација објективној функцији са друге стране.
- PEST има два додатна, напредна, јединствена и изузетно моћна начина рада позната као "режим предвиђене анализе" и "регуларизацијски режим".
- Ако модел може израчунати сопствену осетљивост, параметри се могу испоручити PEST-у; ово може довести до огромног повећања ефикасности оптимизације.
- Паралелни PEST се може користити за дистрибуцију модела кроз мреже; ово може знатно смањити време обраде.
- Ако модел може израчунати сопствене деривате и проследити их PEST-у у датотеку, PEST може користити ове деривате, чиме решава проблем да их мора израчунати коначним разликама. То може значајно повећати ефикасност процеса процјене параметара.
- Нови PEST програм дозвољава кориснику да изврши инспекцију Јакобијанове матрице у било којој фази процеса процене параметара.

На располагању су и PC и UNIX верзије PEST-а. У PC и UNIX верзије PEST-а укључени су бројни корисни програми који помажу у припреми и управљању подацима.

Ако се рачунарски модел користи за разумевање или тумачење података који се односе на систем, велике су шансе да ће учинак модела бити значајно побољшан коришћењем PEST-а. PEST се успешно користи у већини научних области, укључујући хидрологију подземних и површинских вода, геофизику, геомеханику, хемију, ваздухопловство, машинство, биологију и науку о земљишту [5].

## 3.2. PEST улазне датотеке

PEST захтева три типа улазне датотеке. То су:

• Template датотеке - по један за сваку улазну датотеку модела у којој су дефинисани параметри;

• Instruction датотеке - по једна за сваку излазну датотеку модела на којој постоје вредности генерисане моделом

• Контролну датотеку - која садржи PEST са именима свих датотека инструкција, имена одговарајућих улазних и излазних датотека модела, врсту проблема, контролне варијабле, почетне параметре, мерне вредности итд.

У даљем тексту биће описан начин креирања, структура и намена Template и Instruction датотека [6].

### 3.2.1. PEST Template датотеке

Кад год PEST покреће модел, пошто то мора учинити много пута приликом попуњавања Јакобијанове матрице или решавања инверзног проблема, мора прво исписати вредности параметара у улазне датотеке модела у којима су похрањене. Модел се покреће да би израчунао објективну функцију која произилази из вредности почетних параметара добијених од стране корисника, да би се тестирала надоградња параметара, или да се израчунају деривати опсервација у односу на одређени параметар. PEST обезбеђује скуп вредности параметара које су потребне моделу и ће се користити за тај конкретан прорачун. Једини начин на који модел може да приступи овим вредностима је да их прочита из улазних датотека.

Неки модели читају неке или све своје податке са терминала, од којих корисник мора да испоручује ове ставке података у одговору на позивне типове модела. Ово се може учинити и кроз датотеку. Ако пишете у датотеку одговоре који ћете обично доставити моделу преко терминала, можете их "преусмерити" на модел помоћу симбола "<" на моделској наредбеној линији. Дакле, ако се ваш модел покреће помоћу команде

"model", а ви унапред упишете своје одговоре у датотеку file.inp, онда ви (и PEST) можете покренути модел без потребе за уносом терминала помоћу наредбе

model < file.inp

Ако file.inp садржи параметре које PEST мора оптимизовати, може се направити образац за њега као да је било која друга улазна датотека модела.

Модел може читати многе улазне датотеке, међутим, наредба је потребна само за оне улазне датотеке које садрже параметре који захтевају процену. PEST не мора идентификовати ни једну од осталих улазних датотека модела.

Моделна улазна датотека може бити од било које дужине. Међутим PEST инсистира да не буде више од 2000 знакова у ширини. Исто важи и за Template датотеке. Препоручено је да Template датотеке буду сачуване под екстензијом ".tpl" како би их разликовали од других типова датотека [7].

### 3.2.2. PEST Instruction датотеке

Од могућих обимних количина информација које модел може написати у своје излазне датотеке, PEST идентификује само за неколико вредности. То могу бити вредности за које су расположиви одговарајући подаци о пољу или позицији и за које се одступања између излаза модела и измерених вредности морају свести на минимум у пондерисаном смислу најмањих квадрата. Алтернативно, оне могу бити предвиђања модела, или могу једноставно бити излазни модели за које се захтиева осетљивост у односу на параметре. Ови одређени бројеви генерисани моделом се називају "запажања" или "посматрања генерисана моделом".

За сваку излазну датотеку модела која садржи опсервације, морате дати датотеку инструкција која садржи упутства која PEST мора да прати како би прочитало ту датотеку. Треба имати на уму да ако је излазна датотека модела већа од 2000 знакова у ширини, PEST неће моћи да је прочита; међутим модел излазне датотеке може бити било које дужине.

Неки модели пишу неке или све њихове излазне податке на терминал. Можете да преусмерите овај излаз у датотеку помоћу симбола ">" и научите ПЕСТ како да чита ову датотеку помоћу одговарајуће Instruction датотеке на уобичајени начин.

Препоручљиво је да се Instruction датотеке сниме под екстензијом ".ins" како би их разликовали од других типова датотека [7].

# 4. FEMAP – програмски пакет за моделирање и постпроцесирање

FEMAP (Finite Element Modeling And Postprocessing) је програм намењен за инжењерске анализе, компаније Siemens PLM Software који се користи за решавање сложених инжењерских проблема методом коначних елемената (предпроцесирање) и приказ резултата анализе (постпроцесирање). Ради на Microsoft Windows платформи и обезбеђује CAD увоз, алате за креирање модела коначних елемената, алате за креирање мреже, као и функције постпроцесирања која омогућава квалитетну и компактну интерпретацију резултата анализе. Метода коначних елемената омогућава инжењерима да виртуелно моделирају компоненте, склопове или системе за одређивање понашања под датим скупом граничних услова и обично се користе у процесу пројектовања како би се смањила потреба за скупом израдом прототипова и тестирањима, олакшала процијенилна промена дизајна и материјала.

Апликације за симулацију производа укључују базичну анализу јачине, фреквенцију и транзијентну динамичку симулацију, процену перформанси на нивоу система и напредни одзив, ток течности и мулти-физичку инжењерску анализу за симулацију функционалних перформанси [8].

Femap је CAD неутралан, што значи да се може користити без обзира на CAD програм којим се моделира. Има уграђене транслаторе за преузимање геометрије из свих водећих CAD програма, као што су Solid Edge, NX, Pro/E, Catia, Solid Works, Inventor, AutoCAD и других. Такође на располагању су сви потребни програмски алати за модификацију геометрије површина и 3D модела. Припрема геометрије за различите врсте анализа је зато брза и једноставна.

Femap укључује низ програмских алата за аутоматско отклањање непотребних детаља (спољашњих заобљења, мањих рупа, удруживање мањих површина итд.). Код делова са танким зидовима (љуске) или код конструкција од лимова, мрежу је потребно креирати на средњим површинама, које Femap сам креира уз коришћење вредности параметара, са којима се може контролирати процес креирања средњих површина.

Може виртуално анализирати индивидуалне моделе или склопове и одредити њихово понашање као одговор на задате потребе у одређеном радном окружењу.

Укључује оптимизацију default параметара за креирање FE мреже, зависно о типу и облику геометрије. Са додатним параметрима корисник може, већ према властитом искуству код креирања FE мрежа, додатно утицати на квалитет мреже коначних елемената.

Коришћњем Femap-ових дигиталних симулација могу се:

- Предвидети и побољшати перформансе производа и њихова поузданост
- Смањити време испитивања и тестирања, као и избећи скупи физички прототипови
- Проценити квалитет дизајна и материјала
- Оптимизовати конструкције и смањити коришћење материјала [9]

# 5. Метод коначних елемената

Метод коначних елемената (МКЕ) је данас најопштији нумерички метод, примењен у готово свим наукама, а посебно у инжењерским областима. Сматра се да је МКЕ метод који је, у поређењу са другим, имао највећи утицај на развој и домете техничких дисциплина у двадесетом веку. Велики број међународних часописа највишег нивоа, читава индустрија за развој софтвера на основима МКЕ, свакодневна примена МКЕ комерцијалних и специјалних програмских пакета у оквиру CAD система у свим индустријским гранама, и интензивна научна истраживања у МКЕ, најкраће су илустрација претходне тврдње. У развијеним земљама МКЕ курсеви на редовним и последипломским студијама на техничким факултетима су постали део општег образовања студената.

Основне карактеристике МКЕ које га доводе испред других метода се састоје у његовој општости (применљив је на област солида, флуида, провођења топлоте, и уопште на проблеме поља физичких величина, као и на спрегнуте проблеме), применљивости на линеарне и нелинеарне проблеме, релативној једноставности у основним поставкама, као и погодности у погледу примене рачунара.

Као резултат веома интензивног рада у МКЕ и могућности примене у инжењерској пракси, развијен је велики број комерцијалних пакета опште намене, од којих су најпознатији, на пример, NASTRAN, MARC, ANSYS, ADINA, ABAQUS, SAP, и др. Код нас је развијен програм оваквог типа PAK (од: Програм за Анализу Конструкција) [10].

## 5.1. Развој и област примене програма PAK

Развој програма PAK започео је у Институту за аутомобиле и Рачунском центру "Заставе", 1975, прво за област линеарне анализе конструкција, а затим за проблеме нелинеарне анализе (нелинеарно понашање материјала, геометријска нелинеарност, велика померања), провођења топлоте, механику флуида, биомеханику и спрегнуте проблеме поља физичких величина.

PAK је програм који је на нивоу светски познатих и признатих софтверских пакета који служе за структурне анализе, базиран на теоријским поставкама заснованим на методу коначних елемената [10].

## 5.2. Врсте анализа које се могу вршити програмом PAK

Програмски пакет PAK садржи 6 солвера који се користе за различите врсте анализа.

- PAK-S - програм за линеарну и нелинеарну структурну анализу
- PAK-T - програм за линеарну и нелинеарну анализу топлотне кондукције
- PAK-F - програм за анализу ламинарног струјања нестишљивог флуида и пренос топлоте
- PAK-P - програм за анализу протока флуида кроз деформабилне и порозне структуре
- PAK-B - програм за анализе у биомеханици, моделирању мишића и везивних ткива и протока крви кроз артерије
- PAK-FM - програм за анализе механике лома и замора

У даљем раду биће коришћен солвер за структурну анализу, PAK-S.

## 5.3. 3D изопараметарски коначни елемент

Овај коначни елемент се користи за моделирање тродимензионалних тела општег облика (3D континуума). Елемент може имати различит број чворова - уобичајен је број од 8 до 21 и овакав елемент зваћемо основни, који има шест површи које га ограничавају. Међутим, од овог елемента се могу формирати и други просторни облици, као што су просторна призма, тетраедар или четворострана пирамида. Овакви елементи настали поклапањем неких чворова основног елемента, зову се дегенерисани 3D елементи. Побољшање резултата добијених моделирањем 3D елементима са осам чворова, у случају општих услова деформисања, може се постићи увођењем корективних функција за поље деформације у елементу, помоћу тзв. инкомпатибилних померања. Овакви елементи се зову побољшани 3D елементи. Даље излагање је организовано тако што прво излажемо основни 3D елемент, а затим дегенерисани елемент [10].

## 5.4. Основни 3D елемент

Интерполације геометрије и померања су, према (5.1) и (5.2), облика:

$$\mathrm{x} = \begin{Bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \mathrm{x}_3 \end{Bmatrix} = \begin{Bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{Bmatrix} = \mathbf{HX} \tag{5.1}$$

$$\mathrm{u} = \begin{Bmatrix} \mathrm{u}_1 \\ \mathrm{u}_2 \\ \mathrm{u}_3 \end{Bmatrix} = \begin{Bmatrix} \mathrm{u}_\mathrm{x} \\ \mathrm{u}_\mathrm{y} \\ \mathrm{u}_\mathrm{z} \end{Bmatrix} = \mathbf{HU} \tag{5.2}$$

где су интерполациона матрица **H**

$$\mathbf{H} = \begin{bmatrix} h_1 & 0 & 0 & h_2 & 0 & 0 & \cdots\cdots & \mathrm{h_N} & 0 & 0 \\ 0 & h_1 & 0 & 0 & h_2 & 0 & \cdots\cdots & 0 & \mathrm{h_N} & 0 \\ 0 & 0 & h_1 & 0 & 0 & h_2 & \cdots\cdots & 0 & 0 & \mathrm{h_N} \end{bmatrix} \tag{5.3}$$

и вектори координата чворова **X** и померања чворова **У**.

$$\mathbf{X}^T = [X_1^1\ X_2^1\ X_3^1\ X_1^2\ X_2^2\ X_3^2 \ldots\ldots X_1^N X_2^N X_3^N] \tag{5.4}$$

$$\mathbf{U}^T = [U_1^1\ U_2^1\ U_3^1\ U_1^2\ U_2^2\ U_3^2 \ldots\ldots U_1^N U_2^N U_3^N] \tag{5.5}$$

Приметимо да се (4.2.1) и (4.2.2) могу написати у компонентном облику као

$$\mathrm{x_i} = \sum_{k=1}^{N} h_k\, X_i^k \qquad \mathrm{i} = 1,2,3 \tag{5.6}$$

$$\mathrm{u_i} = \sum_{k=1}^{N} h_k\, U_i^k \qquad \mathrm{u} = 1,2,3 \tag{5.7}$$

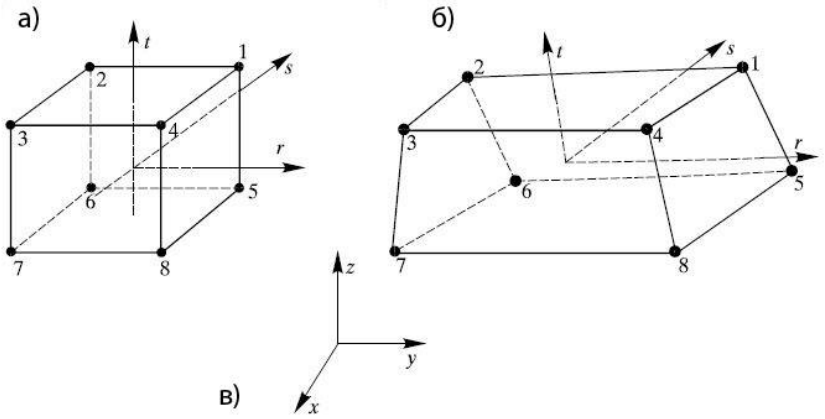
где i =1 одговара оси x, i = 2 - оси y , а i = 3 - оси z . У случају коначног елемента са 8 чворова, интерполационе функције су

$$\begin{aligned} h_1 &= \frac{1}{8}(1+r)(1+s)(1+t) & h_5 &= \frac{1}{8}(1+r)(1+s)(1-t) \\ h_2 &= \tfrac{1}{8}(1-r)(1+s)(1+t) & h_6 &= \tfrac{1}{8}(1-r)(1+s)(1-t) \\ h_3 &= \frac{1}{8}(1-r)(1-s)(1+t) & h_7 &= \frac{1}{8}(1-r)(1-s)(1-t) \\ h_4 &= \frac{1}{8}(1+r)(1-s)(1+t) & h_8 &= \frac{1}{8}(1+r)(1-s)(1-t) \end{aligned} \tag{5.8}$$

односно

$$h_i(r,s,t) = \tfrac{1}{8}(1+rr_i)(1+ss_i)(1+tt_i) \qquad i = 1,2,\ldots\ldots 8 \tag{5.9}$$

Функције $h_i$(r, s,t) су линеарне по r, s, t и зато се овај елемент зове линеаран. Његове површи су, како се види са сл. 5.1, равни у физичком простору. Елемент се из x, y, z простора пресликава на коцку чије су координате чворова $r_i, s_i, t_i$ једнаке ±1, као што је дато на слици 5.1. Приметимо да се усвојени редослед у нумерисању чворова мора користити у практичној примени 3D елемента, да би се очувало једнозначно пресликавање и десна оријентација природног система r, s, t.



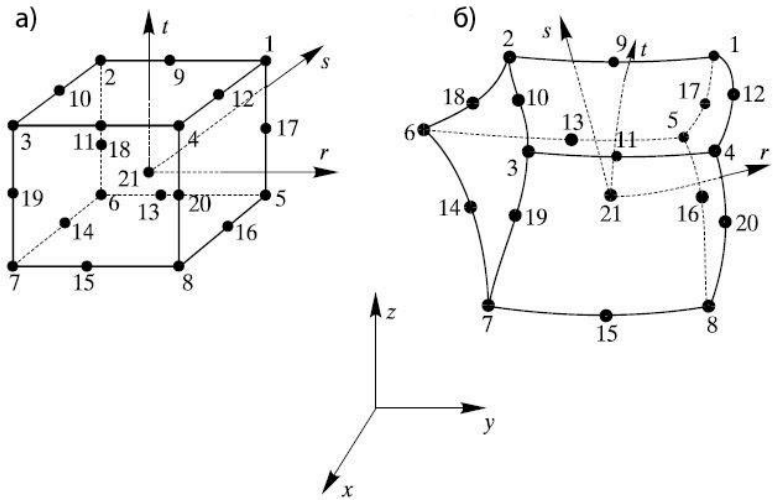

| Чвор | $r_i$ | $s_i$ | $t_i$ |
|---|---|---|---|
| 1 | +1 | +1 | +1 |
| 2 | -1 | +1 | +1 |
| 3 | -1 | -1 | +1 |
| 4 | +1 | -1 | +1 |
| 5 | +1 | +1 | -1 |
| 6 | -1 | +1 | -1 |
| 7 | -1 | -1 | -1 |
| 8 | +1 | -1 | -1 |

*Слика 5.1. 3D коначни елемент са 8 чворова а) Природни координатни систем r,s,t;*

*б) Декартов координатни систем x,y,z (физички простор)*

*в) Вредности природних координата у чворовима елемента [10]*

За моделирање кривих површи могу се користити 3D елементи са више од 8 чворова. На сл. 5.2. показан је 3D елемент са 21 чвором. Површи елемента могу бити закривљене, а елемент се пресликава на коцку чији угаони чворови имају координате ±1 у природном координатном систему r, s, t. Међучворови имају по једну природну координату једнаку нули, како се и са слике види, док се чвор 21 налази у координатном почетку природног система.




*Слика 5.2. 3D елемент са 21 чвором а) Природни координатни систем*

*б) Декартов координатни систем [10]*

Ако се користи нумерација чворова према сл. 5.2., интерполационе функције $h_i$ се могу изразити у облику

$$\begin{aligned} h_1 &= g_1 - \frac{1}{2}(g_9 + g_{12} + g_{17}) & h_6 &= g_6 - \frac{1}{2}(g_{13} + g_{16} + g_{17}) \\ h_2 &= g_2 - \tfrac{1}{2}(g_9 + g_{10} + g_{18}) & h_6 &= g_6 - \tfrac{1}{2}(g_{13} + g_{14} + g_{18}) \\ h_3 &= g_3 - \frac{1}{2}(g_{10} + g_{11} + g_{19}) & h_7 &= g_7 - \frac{1}{2}(g_{14} + g_{15} + g_{19}) \\ h_4 &= g_3 - \frac{1}{2}(g_{11} + g_{12} + g_{20}) & h_8 &= g_8 - \frac{1}{2}(g_{15} + g_{16} + g_{20}) \end{aligned} \qquad (5.10)$$

$$h_k = g_k \qquad k = 9, 10, \ldots\ldots, 20$$

$$h_k = 0 \qquad \text{ako чвор не постоји}$$

где су функције $g_i$ одређене као

$$g_k = G(r, r_k)G(s, s_k)G(t, t_k)$$

$$G(\beta, \beta_k) = \tfrac{1}{2}(1 + \beta_k\beta) \quad \text{за} \quad \beta_k = \pm 1 \qquad \beta = r, s, t \qquad (5.11)$$

$$G(\beta, \beta_k) = 1 - \beta^2 \qquad \text{за} \quad \beta_k = 0$$

Приметимо да се основне интерполационе функције за угаоне чворове, одређене преко (5.8), (5.9), коригују интерплационим функцијама суседних међучворова. Коначно, интерполациона функција $h_{21}$ одређена је као

$$h_{21} = (1 - r^2)(1 - s^2)(1 - t^2) \qquad (5.12)$$

У случају постојања чвора 21 коригују се интерполационе функције $h_i$ дате са према следећем

$$h_k = \overline{h}_k - \tfrac{1}{8} h_{21} \qquad k = 1, 2, \ldots\ldots, 8 \qquad (5.13)$$

$$h_k = \overline{h}_k - \tfrac{1}{4} h_{21} \qquad k = 9, 10, \ldots\ldots, 20$$

где смо са $\overline{h}_k$ означили интерполационе функције одређене једначином.

Интерполационе функције дате са (5.10), односно са (5.13), садрже квадрате природних координата, па се овакви елементи зову и параболични. Интересантно је приметити да интерполационе функције $h_i$, дефинисане било једначином (5.8) или (5.13), имају особину већ истакнуту код елемената штапа и показану на сл. 3.2.4; функција $h_k$ = 1 у чвору " к ", а једнака је нули у свим осталим чворовима.

Дакле

$$h_k = (r_j, s_j, t_j) = \delta_{kj} \qquad (5.14)$$

где је $\delta_{kj}$ Кронекеров делта симбол $(\delta_{кj} = 1,\ k = 1,\ \delta_{kj} = 0,\ k \neq j)$.

Како се види са слике 5.1. и 5.2. површи које ограничавају елемент одређене су условом да је једна од природних координата константа и једнака +1 или -1. На пример,

површи r = −1 одговара површ на којој, према сл. 5.2, леже чворови 2, 6, 7, 3, 18, 14, 19, 10. Координатне линије природног координатног система су одређене условом да су две природне координате једнаке нули. На пример, координатна r - линија одређена је са s = t = 0; на слици 5.2а то је права, а на слици 5.2б то је крива линија.

У принципу, елементи вишег реда дају тачније резултате за опште случајеве поља деформације и напона, и сложене геометријске облике тела. Међутим, моделирање помоћу оваквих елемената је сложеније, па се користе линеарни елементи, али са гушћом мрежом. Такође, значајно је побољшање тачности линеарних елемената помоћу инкомпатибилних померања.

Изведимо сада матрицу **B** ради одређивања деформације према (5.3). У складу са дефиницијом деформације е према (5.4) и интерполацијама за померање (5.2), односно (5.7), имамо да је

$$e = \begin{Bmatrix} \mathrm{e_{xx}} \\ \mathrm{e_{yy}} \\ \mathrm{e_{zz}} \\ \mathrm{\gamma_{xy}} \\ \mathrm{\gamma_{yz}} \\ \mathrm{\gamma_{zx}} \end{Bmatrix} = \begin{Bmatrix} \mathrm{u_{1,1}} \\ \mathrm{u_{2,2}} \\ \mathrm{u_{3,3}} \\ \mathrm{u_{1,2}+u_{2,1}} \\ \mathrm{u_{2,3}+u_{3,2}} \\ \mathrm{u_{1,3}+u_{3,1}} \end{Bmatrix} = \begin{bmatrix} \mathrm{h_{1,1}} & 0 & 0 & \cdots & \mathrm{h_{N,1}} & 0 & 0 \\ 0 & \mathrm{h_{1,2}} & 0 & \cdots & 0 & \mathrm{h_{N,2}} & 0 \\ 0 & 0 & \mathrm{h_{1,3}} & \cdots & 0 & 0 & \mathrm{h_{N,3}} \\ \mathrm{h_{1,2}} & \mathrm{h_{1,1}} & 0 & \cdots & \mathrm{h_{N,2}} & \mathrm{h_{N,1}} & 0 \\ 0 & \mathrm{h_{1,3}} & \mathrm{h_{1,2}} & \cdots & 0 & \mathrm{h_{N,3}} & \mathrm{h_{N,2}} \\ \mathrm{h_{1,3}} & 0 & \mathrm{h_{1,1}} & \cdots & \mathrm{h_{N,3}} & 0 & \mathrm{h_{N,1}} \end{bmatrix} \begin{Bmatrix} U_1^1 \\ U_2^1 \\ U_3^1 \\ \vdots \\ U_1^N \\ U_2^N \\ U_3^N \end{Bmatrix} = \mathbf{BU} \quad (5.15)$$

Одакле се очигледно виде чланови $\mathrm{B_{ij}}$ матрице B. Ради краћег писања увели смо означавање извода као

$$\mathrm{h_{kj}} = \frac{\partial \mathrm{h_k}}{\partial \mathrm{x_j}} \quad (5.16)$$

Овде је потребно детаљније показати начин рачуњања извода $h_{kj}$. Наиме, $\mathrm{h_k}$ су функције природних координата, r,s,t , па се мора применити поступак посредног диференцирања

$$h_{kj} = \frac{\partial h_k}{\partial r}\frac{\partial r}{\partial x_j} + \frac{\partial h_k}{\partial s}\frac{\partial s}{\partial x_j} + \frac{\partial h_k}{\partial t}\frac{\partial t}{\partial x_j} \quad (5.17)$$

Ако се уведе јакобијан трансформације између Декартовог и природног система, као

$$\mathbf{J} = \left[\frac{\partial \mathbf{x}}{\partial \mathbf{r}}\right] = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix} \quad (5.18)$$

а инверзни јакобијан

$$\mathbf{J^{-1}} = \left[\frac{\partial \mathbf{r}}{\partial \mathbf{x}}\right] = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial s}{\partial x} & \frac{\partial t}{\partial x} \\ \frac{\partial r}{\partial y} & \frac{\partial s}{\partial y} & \frac{\partial t}{\partial y} \\ \frac{\partial r}{\partial z} & \frac{\partial s}{\partial z} & \frac{\partial t}{\partial z} \end{bmatrix} \quad (5.19)$$

онда се изводи (5.16) могу написати у облику

$$\mathrm{h_{kj}} = \mathrm{J_{j1}^{-1}}\frac{\partial \mathrm{h_k}}{\partial \mathrm{r}} + \mathrm{J_{j2}^{-1}}\frac{\partial \mathrm{h_k}}{\partial \mathrm{s}} + \mathrm{J_{j3}^{-1}}\frac{\partial \mathrm{h_k}}{\partial \mathrm{t}} \tag{5.20}$$

Изводи $\partial \mathrm{h_k}/\partial \mathrm{r}$, $\partial \mathrm{h_k}/\partial\, \mathrm{s}$, $\partial \mathrm{h_k}/\partial \mathrm{t}$ се добијају диференцирањем које директно следи из дефиниције $h_k(r, s\, t)$ према одговарајућим изразима датим једначинама (5.8),(5.10), (5.13). Чланови јакобијана $\mathrm{J_{mn}}$ следе из дефиниције (5.18) и интерполације за геометрију (5.6),

$$\mathrm{J_{mn}} = \sum_{\mathrm{k=1}}^{\mathrm{N}} \frac{\partial \mathrm{h_k}}{\partial \mathrm{r_m}} \mathrm{X_n^k} \tag{5.21}$$

где је $\mathrm{r_1} = \mathrm{r}$, $\mathrm{r_2} = \mathrm{s}$, $\mathrm{r_3} = \mathrm{t}$.

Горњи поступак одређивања извода функција $h_k$ по Декартовим координатама $h_i$ применљив је на било коју функцију природних координата. Дакле, вектор извода $\partial/\partial x$, дефинисан као

$$\left(\frac{\partial}{\partial \boldsymbol{x}}\right)^T = \left[\frac{\partial}{\partial x} \;\; \frac{\partial}{\partial y} \;\; \frac{\partial}{\partial z}\right] \tag{5.22}$$

можемо матрично изразити релацијом

$$\frac{\partial}{\partial \mathbf{x}} = \mathbf{J}^{-1}\frac{\partial}{\partial \mathbf{r}} \tag{5.23}$$

где је

$$\left(\frac{\partial}{\partial \boldsymbol{r}}\right)^T = \left[\frac{\partial}{\partial r} \;\; \frac{\partial}{\partial s} \;\; \frac{\partial}{\partial t}\right] \tag{5.24}$$

И обрнуто, ако имамо неку функцију координата $h_i$, изводи у односу на природне координате су

$$\frac{\partial}{\partial \mathbf{r}} = \mathbf{J}\frac{\partial}{\partial \mathbf{x}} \tag{5.25}$$

У практичној примени у МКЕ, вредност чланова $\mathrm{J_{mn}}$ јакобијана упосматраној тачки елемента (за одређене вредности природних координата r, s, t), одређује се нумерички а затим се врши инверзија матрице **J** . Наравно, инверзија је могућа за случај несингуларне матрице **J** . До сингуларности може доћи у случају 3D елемента са угловима између граничних површи елемента већим од 180°.

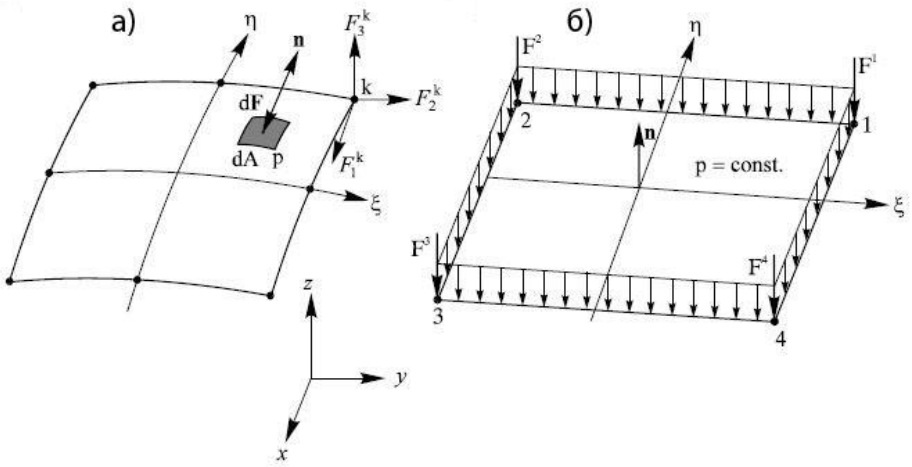
Матрица еластичности C одговара 3D условима и дата је једначином (5.12). Дакле, и вектор напона σ има шест компоненти, у складу са редоследом компоненти деформације e у једначини (5.15). Приметимо да матрица крутости елемента **K** има димензије 3N x 3N, тј.

$$\underset{3\mathrm{Nx3N}}{\mathbf{K}} = \int_{\mathrm{V}} \underset{3\mathrm{Nx6}}{\mathbf{B}^{\mathrm{T}}}\; \underset{6\mathrm{x6}}{\mathbf{C}}\; \underset{6\mathrm{x3N}}{\mathbf{B}}\, \mathrm{dV} \tag{5.26}$$

а вектор сила у чворовима $\mathbf{F}^{\mathrm{u}}$ има 3N компоненти

$$\underset{3\mathrm{Nx1}}{\mathbf{F}^{\mathrm{u}}} = \int_{\mathrm{V}} \underset{3\mathrm{Nx6}}{\mathbf{B}^{\mathrm{T}}}\; \underset{6\mathrm{x1}}{\boldsymbol{\sigma}}\, \mathrm{dV} = \underset{3\mathrm{Nx3N}}{\mathbf{K}}\; \underset{3\mathrm{Nx1}}{\mathbf{U}} \tag{5.27}$$

На крају разматрања основног 3D елемента покажимо начин одређивања еквивалентних сила у чворовима које одговарају површинском притиску на површини елемента. Претпостављамо да на површи чији су чворови 1, 2,....,K делује површински притисак p , како је шематски приказано на сл. 4.2.3.




*Слика 5.3. Површински притисак на страници 3D елемената*

*а) Општи случај, б) Униформни случај на равној правоугаоној страници [10]*

У случају опште криве површи имамо да је елементарна сила притиска dF, која одговара притиску p (ξ, η) у тачки површи одређеној координатама ξ,η ,

$$\mathrm{d}\mathbf{F} = -\mathrm{p}\mathbf{n}\mathrm{dA} \tag{5.28}$$

где је **n** нормала у тачки површи, а dA елементарна површ. Еквивалентне силе се одређују на основу једнакости виртуалних радова притиска и сила у чворовима површи,

$$\sum_{\mathrm{k}=1}^{\mathrm{K}}\left(\mathrm{F}_1^{\mathrm{k}}\delta\mathrm{U}_1^{\mathrm{k}} + \mathrm{F}_2^{\mathrm{k}}\delta\mathrm{U}_2^{\mathrm{k}} + \mathrm{F}_3^{\mathrm{k}}\delta\mathrm{U}_3^{\mathrm{k}}\right) = -\int_{\mathrm{A}} \delta\mathrm{u}_{\mathrm{n}}\,\mathrm{pdAA} \tag{5.29}$$

где је $\delta u_n$ виртуално померање у правцу нормале. Померање неке тачке на површи можемо изразити у облику (5.7), узимајући једну од природних координата једнаку +1 или -1. Овим се функције $h_k(r, s\ t)$ своде на функције две природне координате ξ и η , како је назначено на сл. 5.3. Ове функције следе из 3D интерполационих функција. Дакле, користећи 2D интерполационе функције које одговарају чворовима к =1,.....,K на површи, можемо писати

$$\delta\mathrm{u}_{\mathrm{n}} = \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{h}_{\mathrm{k}}\left(\mathrm{n}_1^{\mathrm{k}}\delta\mathrm{U}_1^{\mathrm{k}} + \mathrm{n}_2^{\mathrm{k}}\delta\mathrm{U}_2^{\mathrm{k}} + \mathrm{n}_3^{\mathrm{k}}\delta\mathrm{U}_3^{\mathrm{k}}\right) \tag{5.30}$$

где су $\mathrm{n}_1^{\mathrm{k}}$, $\mathrm{n}_2^{\mathrm{k}}$ i $\mathrm{n}_3^{\mathrm{k}}$ и пројекције нормале $n^k$ у чвору k на осе x, y, z. Заменом (5.30) у (5.29) и изједначавањем коефицијената уз виртуална померања $\delta U_i^k$, добијамо

$$\mathrm{F}_{\mathrm{i}}^{\mathrm{k}} = -\int_{\mathrm{A}} \mathrm{ph}_{\mathrm{k}}\mathrm{n}_{\mathrm{i}}^{\mathrm{k}}\,\mathrm{dA} \qquad\qquad \mathrm{i} = 1, 2, 3 \tag{5.31}$$

Одређивање површинског интеграла врши се у општем случају нумерички, како је изложено у глави 8. Приметимо да се обично притисак задаје у чворовима, па се вредност у тачки површи израчунава интерполацијом, тј.

$$\mathrm{p} = \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{h_k}\, \mathrm{P^k} \tag{5.32}$$

где је $P^k$ притисак у чвору к . У случају униформног притиска на правоугаоној површи моделираној са 4 чвора, еквивалентне силе су управне на површи и једнаке четвртини укупне површинске силе, како је назначено на слици 5.3.б [10].

## 5.5. Материјалне карактеристике модела

У примерима FEM модела на којима ће се у даљем раду вршити идентификација параметара, решава се систем

$$\boldsymbol{K} * \boldsymbol{U} = \boldsymbol{F} \tag{5.33}$$

где је К – матрица крутости елемента, U – вектори померања чворова елемента.

Матрица крутости елемента се може изразити у следећем облику

$$\mathbf{K} = \int_{\mathrm{V}} \mathbf{B}^{\mathrm{T}} \mathbf{C} \mathbf{B} \mathrm{dV} \tag{5.34}$$

где је С – конститутивна матрица.

У примеру који следи биће вршена идентификација материјалних параметара (Јунгов модул еластичности и Поасонов коефицијент). У примеру је материјал модела линеатан тако да су Е и V садржани у еластичној конститутивној матрици.

$$C^E = [\quad]_{6x6} \leftarrow E, V \tag{5.35}$$

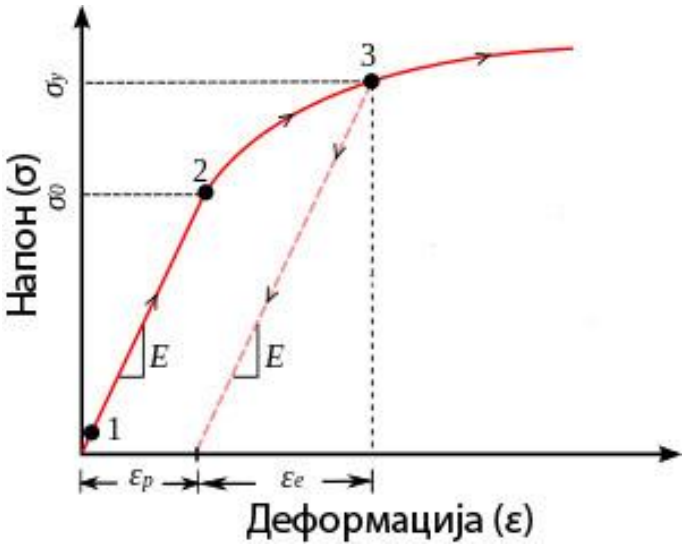
У овом случају, материјал се деформише само еластично, а функција која описује деформацију је линеарна.

При свакој итерацији PEST задаје нове вредности за Е и V и пореди их са референтном вредношћу померања чворова $U^0$. На крају сваке итерације, добија се тренутна вредност померања чворова ΔU, на основу које PEST одређује на који начин коригује тражене параметре (смањује их или повећава).

Након тога, биће вршена идентификација параметара модела чији је материјал нелинеаран. У овом случају, функција деформација у зависности од напона описана је *Ramberg-Osgood*-овом једначином.

$$\sigma = \sigma_y + C_y \varepsilon_p^n \tag{5.36}$$

За разлику од претходног случаја где материјал линеаран у овом случају постоје пластичне деформације $\varepsilon_p$.




*Слика 5.4. Крива напон-деформација добијена Ramberg-Osgood-овом једначином [14]*

На слици 5.4. може се идентификовати зависност поаста еластичних деформација $\varepsilon_e$ од пораста пластичних деформација $\varepsilon_p$.

У овом случају је C еласто-пластична конститутивна матрица. Материјални параметри који ће бити идентификовани карактеристични су за материјал у зони пластичности а то су: напон течења (TAUY), модул пластичности (Cy) и експонент у *Ramberg-Osgood*-овој једначини (AN).

# 6. Идентификација параметара модела оптимизационим процесом у програмском пакету PEST

У даљем раду детаљно ће бити објашњен поступак идентификације одређених параметара модела оптимизационим процесом. У оваквом процесу, на основу референтних излазних вредности добијају се вредности тражених параметара модела.

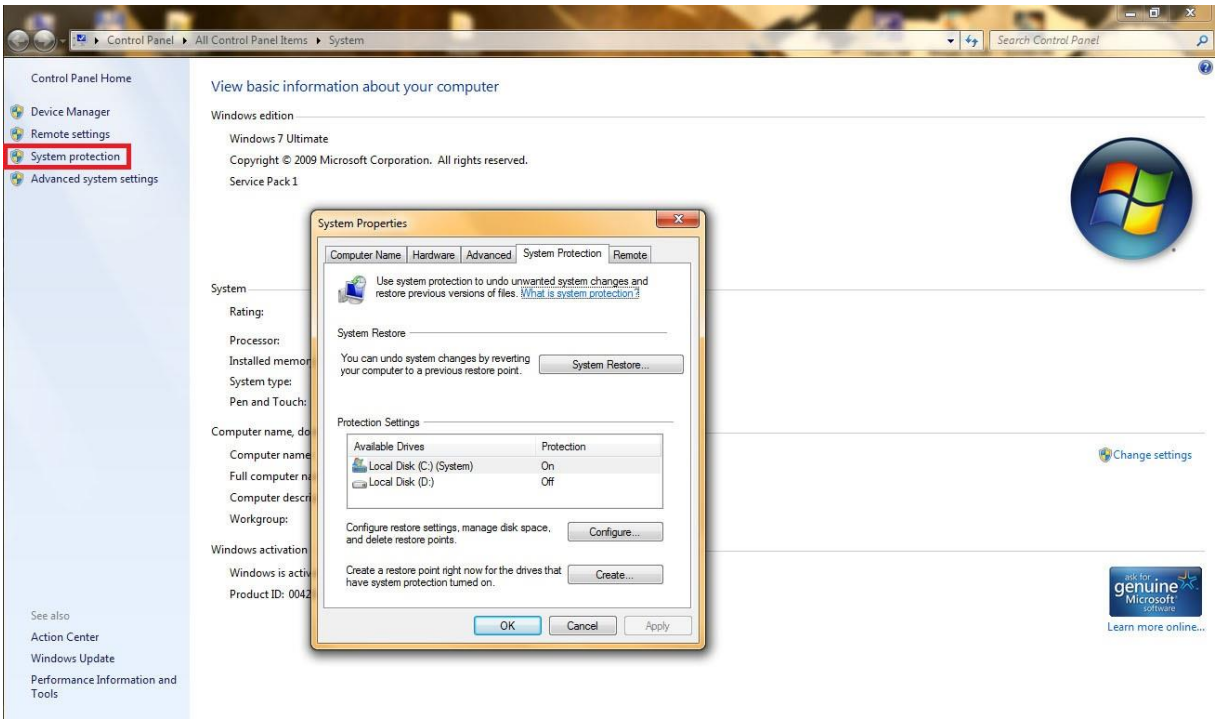
Програмском пакету PEST за овакав прорачун потребан је сет улазних датотека (template files, instruction files), који треба да "научи" PEST које параметре и на основу чега идентификује. Template сет датотека служи да маркира вредности параметара у улазној датотеци који се траже. На овај начин се може маркирати више вредности које треба идентификовати истим оптимизационим прорачуном. Instruction сет датотека се креира како би се оптимизационом прорачуну задале референтне вредности излаза којима се тежи променама вредности циљаних параметара у улазу.

PEST тражене параметре нлази тако што покреће извршни програм а у улазној датотеци мења вредности параметара који се траже. При сваком новом покретању PEST се приближава траженој вредности параметара. На крају сваког прорачуна који покрене PEST пореди тренутне вредности излаза са референтним референтним вредностима које су задате. На тај начин бира вредност параметара које ће уврстити у улазну датотеку приликом нове итерације. Када се добију вредности параметара које се могу сматрати задовољавајућим, прорачун се зауставља. Извештај оптимизационог прорачуна као и добијене вредности параметара штампају се у излазним датотекама које PEST креира.

Један од видова примене овакве идентификације параметара је тај што се на реалном моделу могу добити експериментални резултати одређеног мерења. Ако су неки од параметара модела или процеса непознати, биће потребан велики број различитих мерења како би се они утврдили. Због уштеде времена и ресурса могуће је применити овакву врсту идентификације параметара модела. Потребно је креирати модел у неком од софтверских пакета по узору на реални модел. У PEST-у је потребно маркирати параметре који су непознати и као референтне излазне вредности поставити резултате мерења са реалног модела. Ако су модел и потребне PEST улазне датотеке креиране идентификовани параметри модела поклапаће се са реалним.

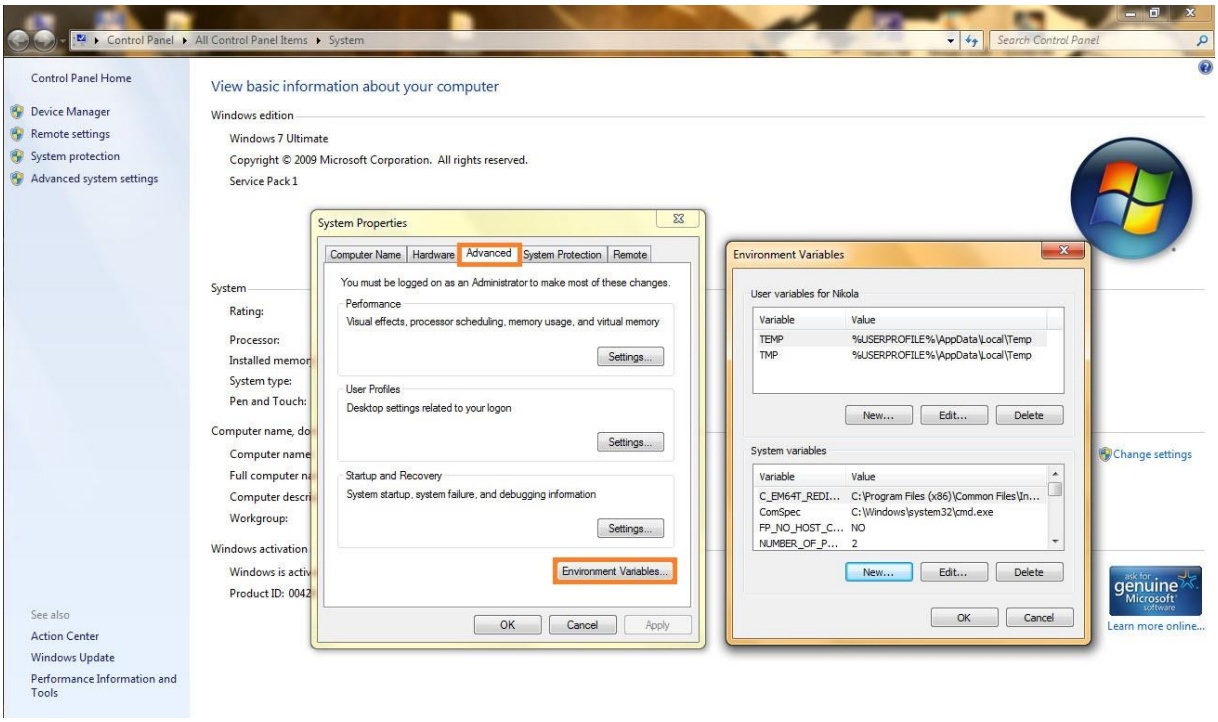
## 6.1. Инсталирање програмског пакета PEST

Пре него што се започне са оптимизацијом потребно је инсталирати програмски пакет PEST. Инсталација се врши веома једноставно. Потребно је све извршне програме који се испоручују са PEST-ом копирати у директоријум који је креиран за ту намену (у овом примеру названом једноставно PEST који се налази на Desktop-у). Затим, потребно је изабрати Control Panel > System > System protection (слика 6.1.).



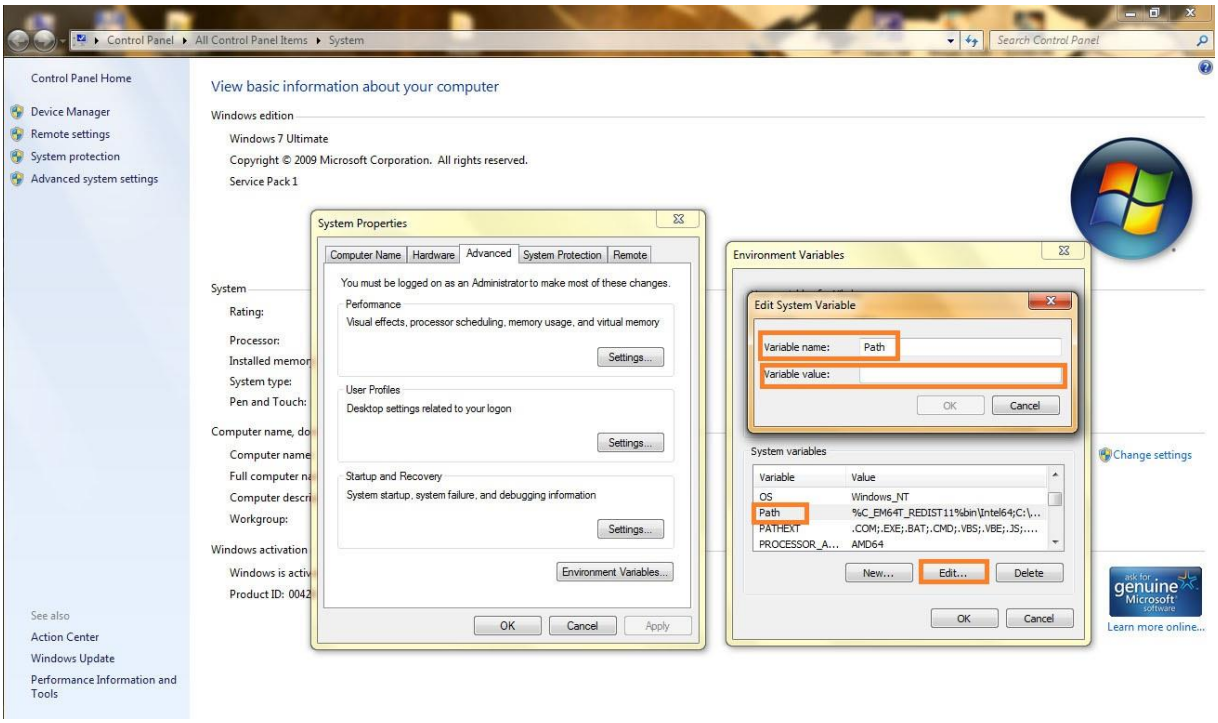

*Слика 6.1. Подешавање у прозору System Properties*

У прозору који се отворио потребно је отворити картицу Advanced, а затим изабрати Environment Variables (слика 6.2.).




*Слика 6.2. Подешавање у прозору Environment Variables*

Након тога, потребно је наћи Path у колони за Variable, означити Path и након тога изабрати Edit. Кликом на Edit отвара се нови прозор у ком је Variable name – Path, а на месту резервисаном за Variable value потребно је унети путању до директоријума у ком су смештени извршни програми PEST-а (у конкретном случају потребно је унети путању до датотеке PEST на desktop-у) (слика 6.3.).




*Слика 6.3. Уништење локације на којој је смештен PEST*

Ова подешавања је потребно сачувати, и на тај начин су извршена подешавања којима се рачунару дозвољава да пронађе извршне програме PEST-а без обзира на тренутни радни директоријум.

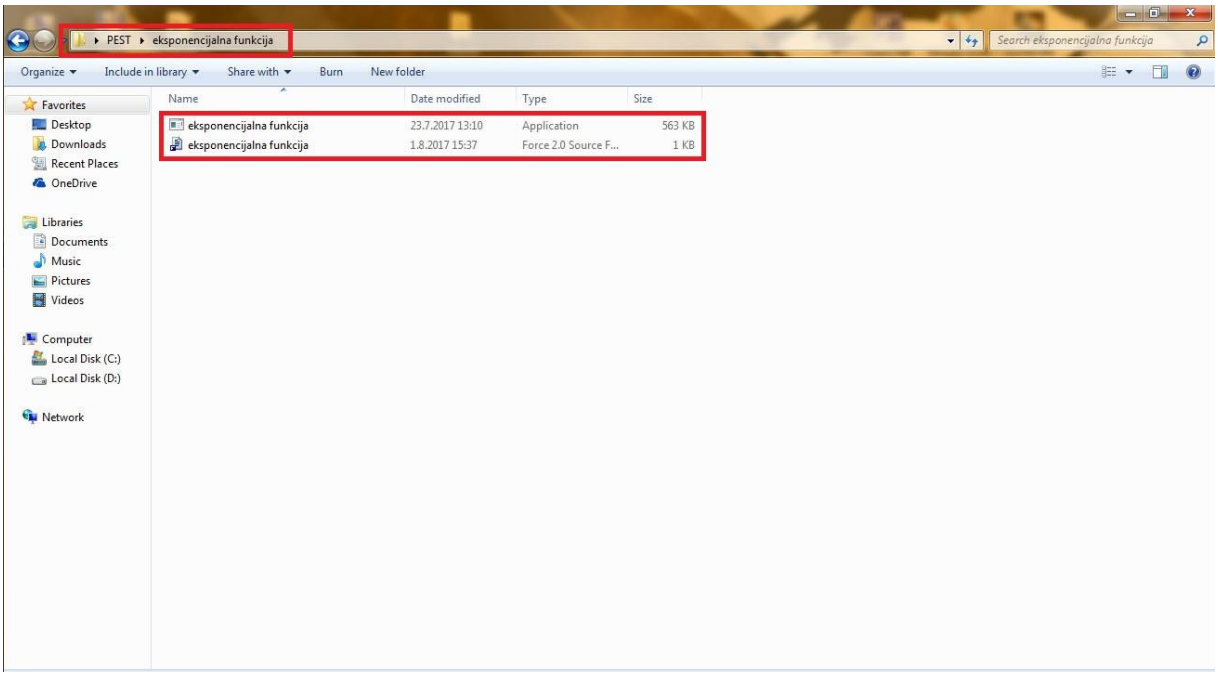
У овом примеру приказана је инсталација PEST-а када се ради под оперативним системом Windows 7. За све остале Windows оперативне системе принцип је веома сличан уз мале разлике у корацима узроковане другачијим изгледом оперативног система [11].

## 6.2. Идентификација параметара експоненцијалне функције

eksponencijalna funkcija.exe је програм који за низ од 13 улазних вредности за *x*, експоненцијалном функцијом $y = x^{x_c} + y_1$, израчунава вредности *y*. Програм eksponencijalna funkcija је писан у програмском језику FORTRAN.

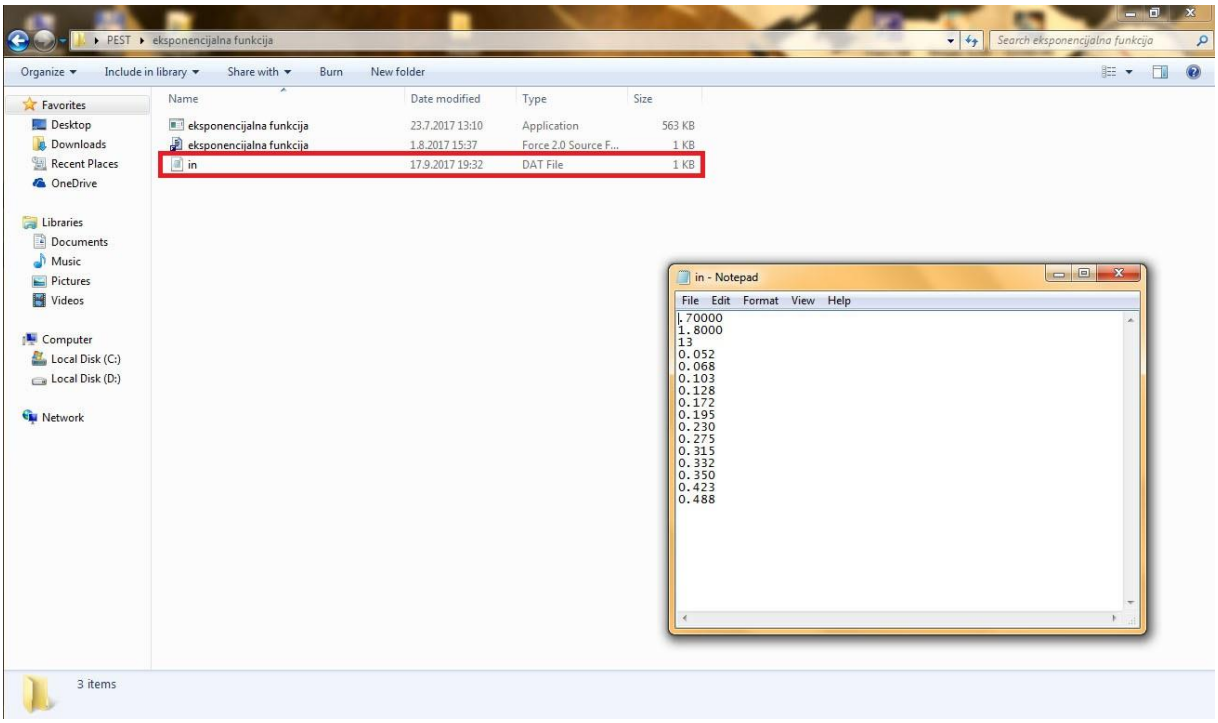
Циљ оптимизације је са што мањом грешком израчунати параметре експоненцијалне функције $x_c$ и $y_1$, ако су дефинисане улазне вредности за *x*, а за излазне вредности узму се вредности добијене када се програм eksponencijalna funkcija изврши са измењеним параметрима $x_c$ и $y_1$. Циљ овакве оптимизације је погодити са којим вредностима $x_c$ и $y_1$ су добијене излазне вредности које се узимају као референтне при оптимизацији.

Прво је потребно копирати програм eksponencijalna funkcija.exe у директоријум (у овом случају назван eksponencijalna funkcija) који је креиран унутар директоријума PEST у ком се налазе извршни програми (слика 6.4.).

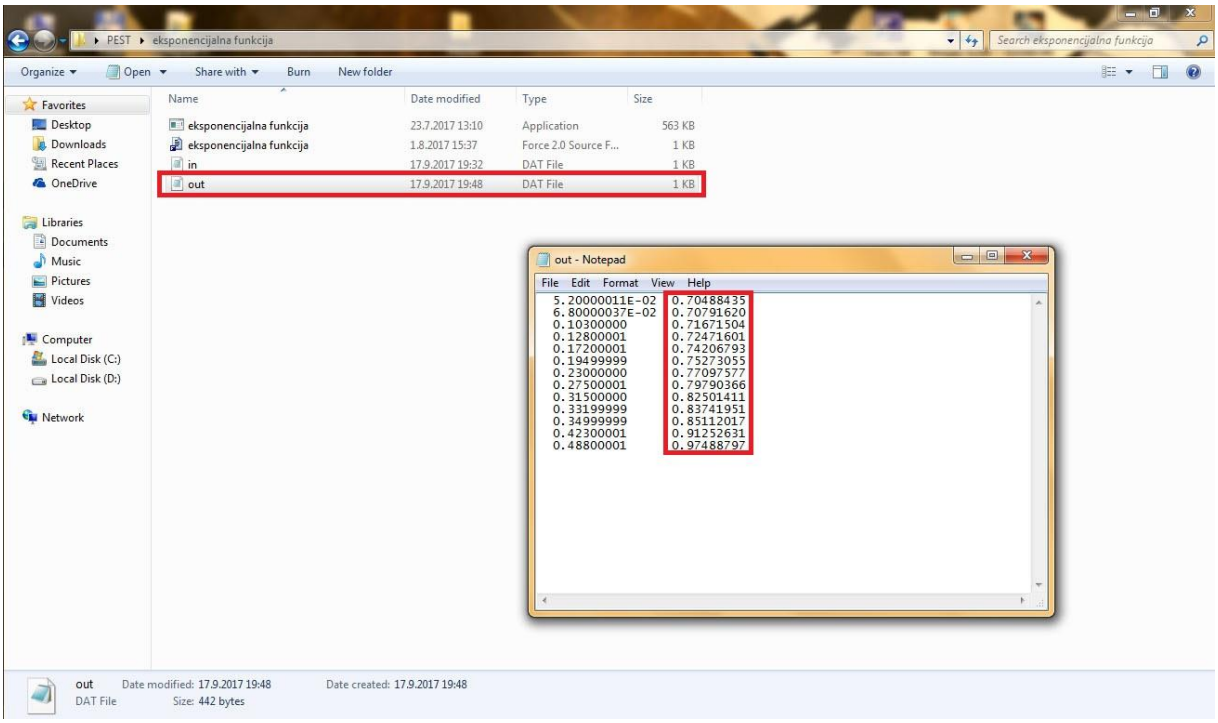



*Слика 6.4. Програм eksponencijalna funkcija смештен у директоријум унутар PEST-а*

Затим, потребно је креирати улазну датотеку in.dat, у којој се налазе вредности параметара $x_c$ и $y_1$ и низ од 13 вредности за *x* као што је приказано на слици. Ова датотека креирана је у notepad-у и снимљена под екстензијом .dat као што је у програму eksponencijalna funkcija дефинисано (слика 6.5.).




*Слика 6.5.Креирање улазне датотеке програма eksponencijala funkcija*

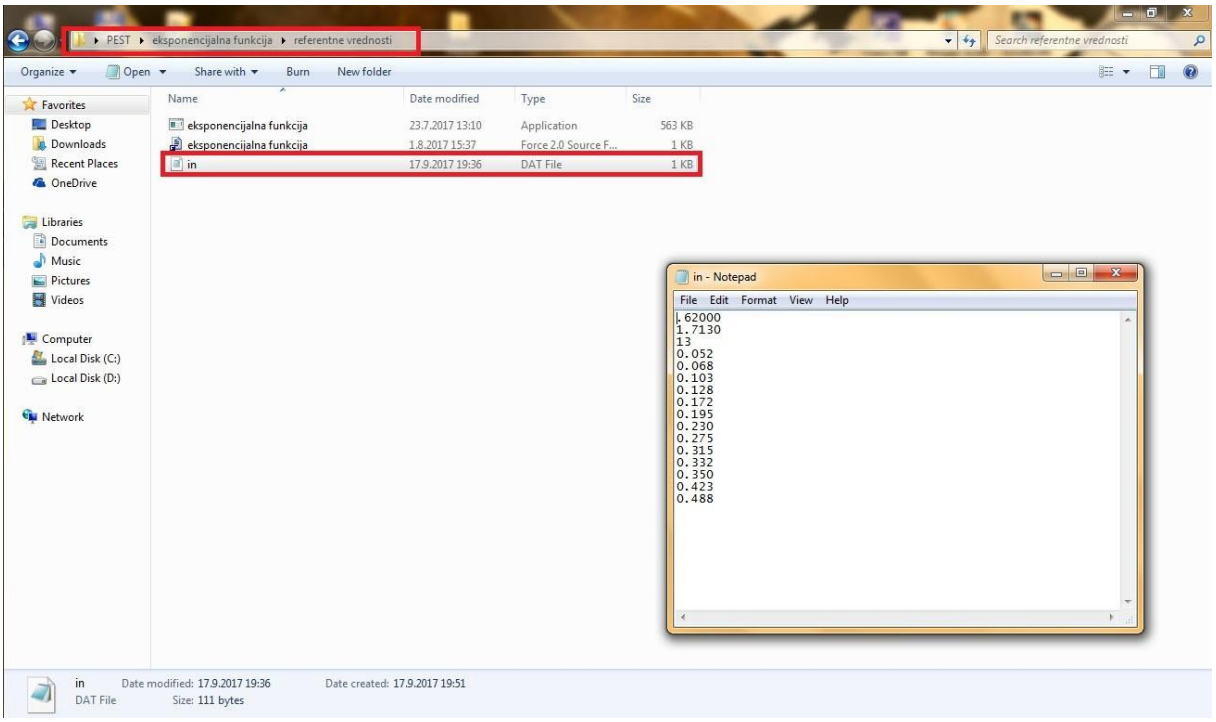



*Слика 6.6. Структура излазне датотеке програма eksponencijalna funkcija*

Када је улазна датотека креирана, потребно је извршити програм (у конкретном случају једноставним покретањем exe-а), при чему ће се аутоматски креирати излазна датотека out.dat. На слици 6.6. је приказана структура излазне датотеке где је обележен низ добијених вредности за *y*.

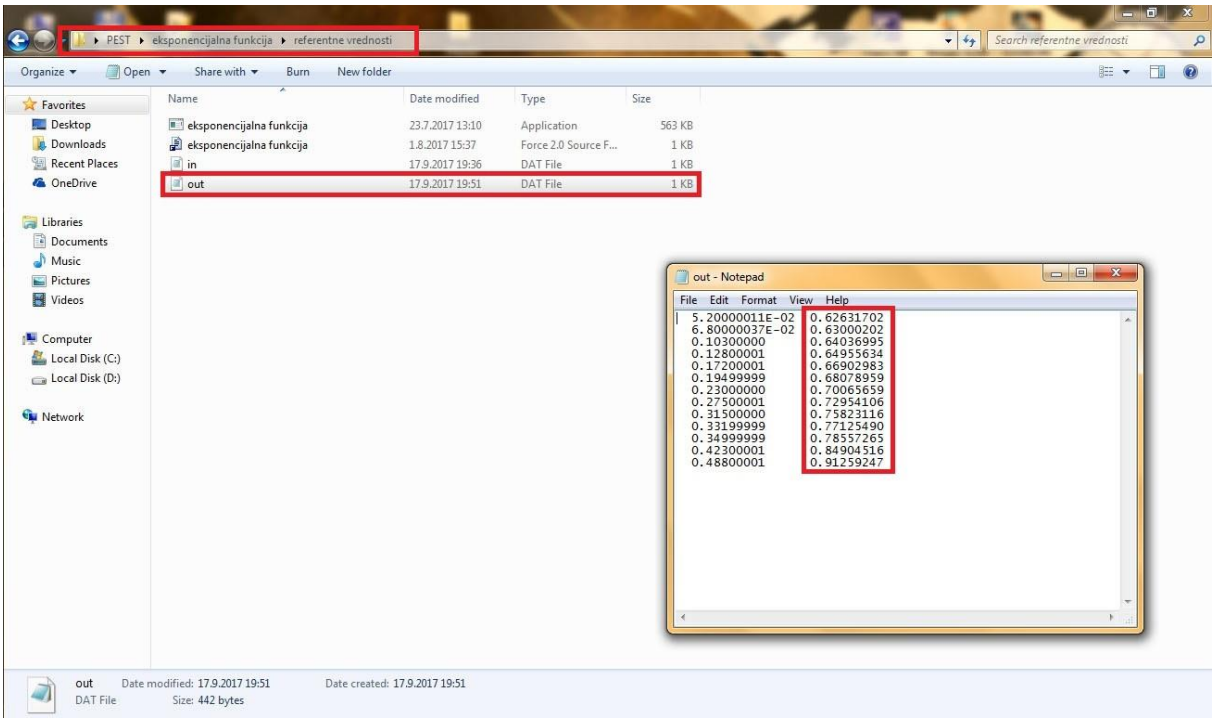
Да би се добиле излазне вредности које би се сматрале референтним и за које би се оптимизацијом тражиле вредности параметара $x_c$ и $y_1$, потребно је у засебном директоријуму извршити програм са истим улазним вредностима за x, али измењеним вредностима за $x_c$ и $y_1$. Вредности $x_c$ и $y_1$ које се овде задају требају да буду нађене у оптимизацији.

У оквиру директоријума eksponencijalna funkcija креиран је нови директоријум referentne vrednosti, где је копиран програм eksponencijalna funkcija. Креирана је улазна датотека са вредностима за *x* идентичним као пре, али са измењеним параметрима $x_c$ и $y_1$. Као што се види на слици 6.7. уместо вредности за $x_c = 0.7$ и $y_1 = 1.8$, задате су вредности $x_c = 0.62$ и $y_1 = 1.713$.




*Слика 6.7. Програм eksponencijalna funkcija извршен у директоријуму referentne vrednosti, са коригованим вредностима параметара у улазној датотеци*

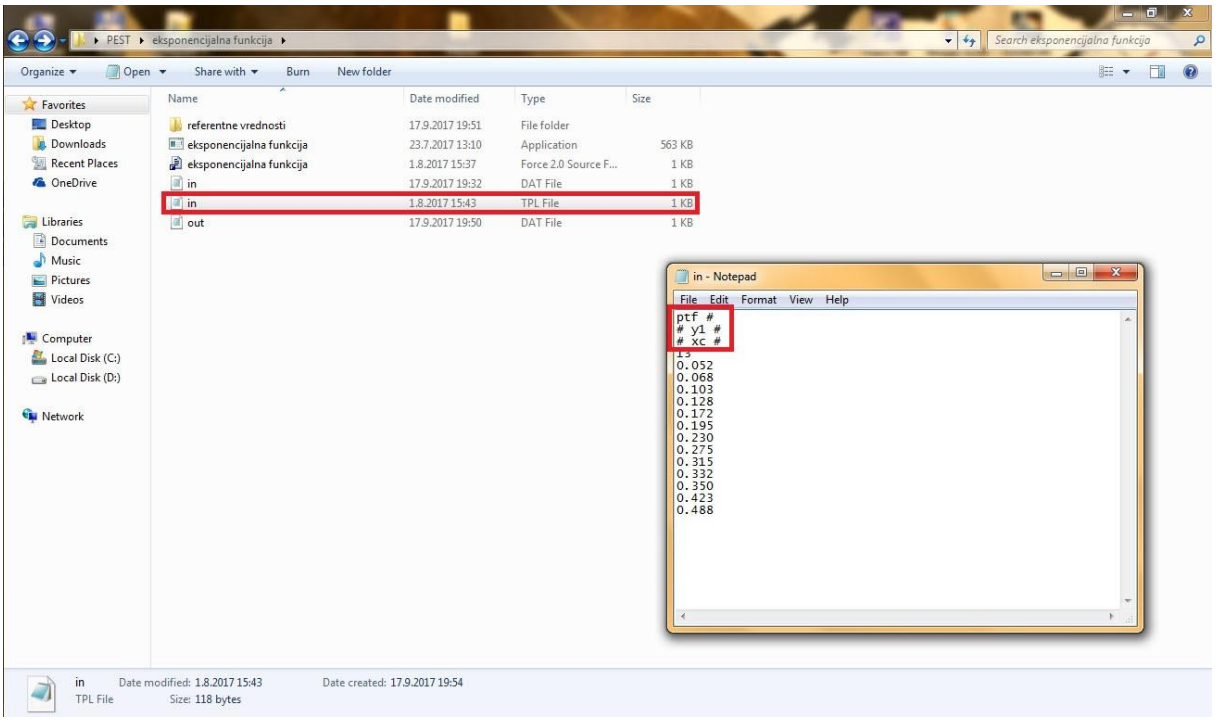
Када се програм изврши добијају се излазне вредности различите од предходних као што је обележено на слици 6.8. Ове вредности ће бити коришћене у оптимизацији, и преко њих PEST треба да дође до вредности параметара $x_c$ и $y_1$.




*Слика 6.8. Структура излазне датотеке када су у улазу измењени параметри*

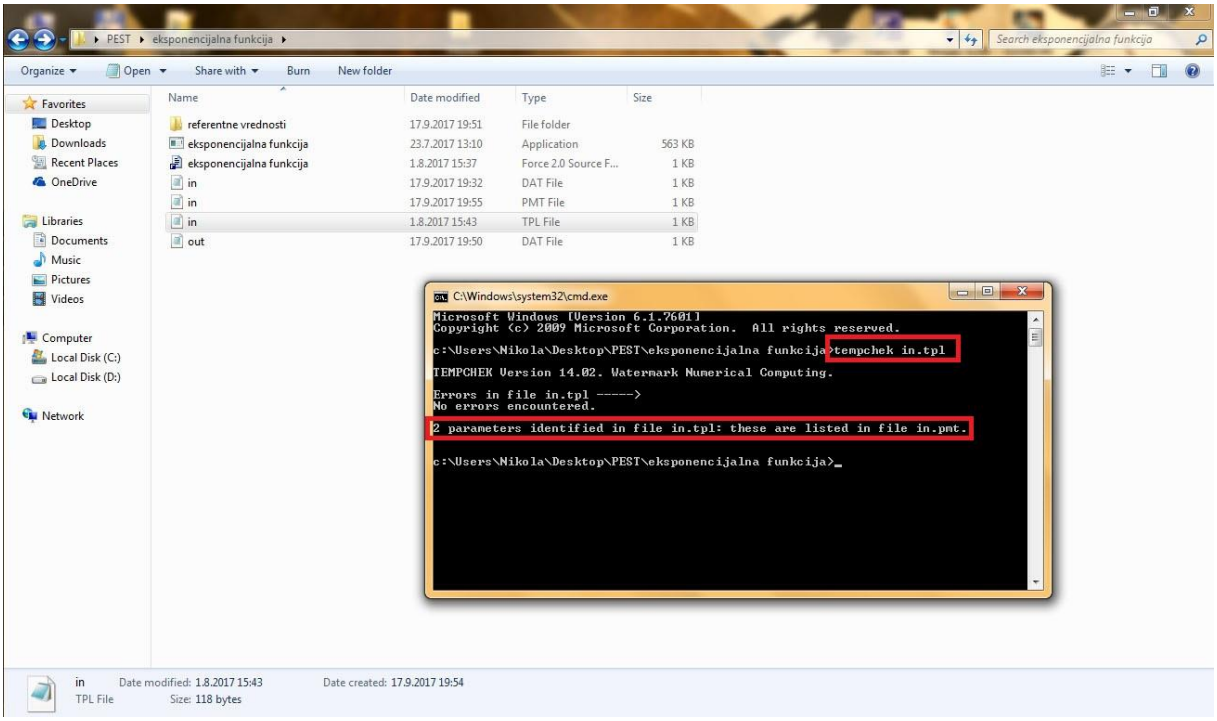
Претходни кораци су били само припрема улазних и излазних датотека које ће се даље користити. Надаље ће бити приказано како креирати низ датотека које ће PEST-у давати инструкције које параметре и на основу чега оптимизовати.

Прво је потребно креирати PEST Template File. Структура Темплате датотеке је скоро у потпуности иста као код улазне датотеке, с тим што Template датотека у првој линији мора садржати инструкцију ptf #. ptf је скраћеница од PEST Template File. На овај начин PEST препознаје Template датотеку и са њом даље манипулише. # представља симбол којим ће испод у датотеци бити означени параметри који ће се варирати и на тај начин бити подвргнути оптимизацији. Позиције на којима су у улазној датотеци биле смештене вредности параметара $x_c$ и $y_1$, овде је потребно маркирати симболом #, а уместо вредности параметра, уписати његову ознаку, $x_c$ и $y_1$ (слика 6.9.). Овако креирану датотеку потребно је сачувати под екстензијом .tpl.



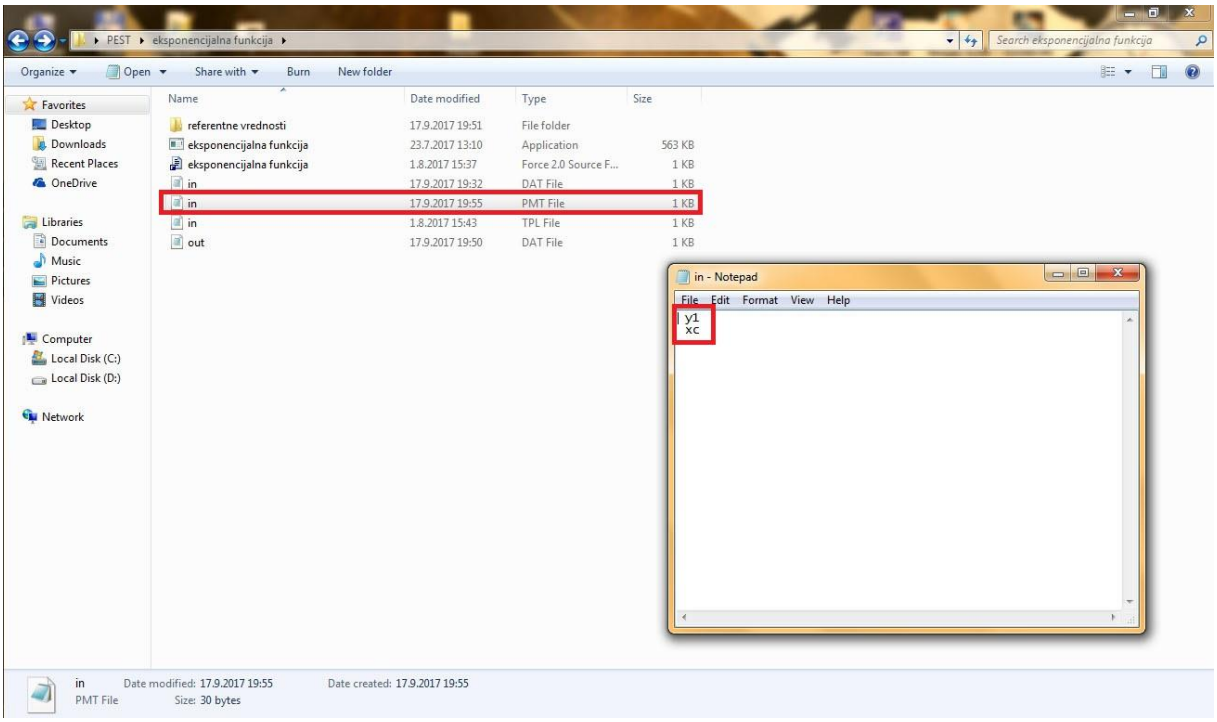

*Слика 6.9. Структура Template датотеке*

Следеће што је потребно урадити јесте проверити да ли је Template датотека коректно креирана. То се ради тако што се покрене command prompt у терминалу где је смештена Temlate датотека и у њему се изда команда pestchek in.tpl (слика 6.10.).

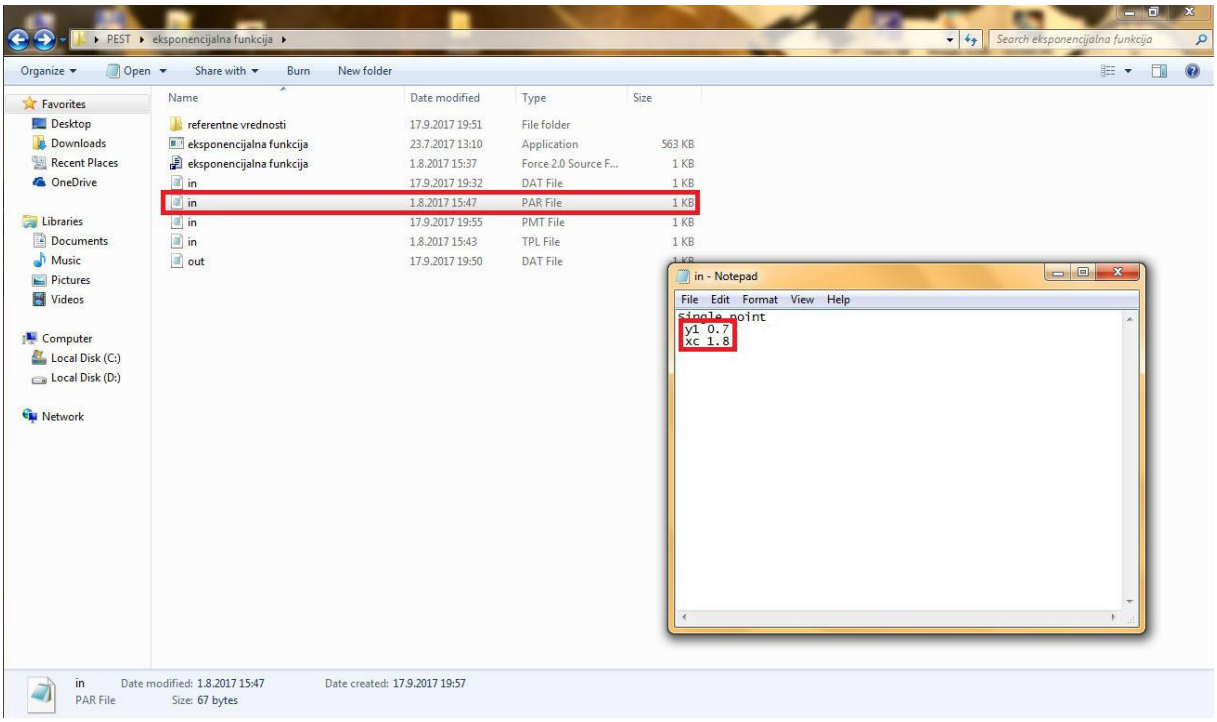



*Слика 6.10. Провера исправности Template датотеке*

Уколико је Template директоријум коректно креиран аутоматски ће бити креирана датотека in.pmt у којој ће бити излистани параметри који су маркирани у Template датотеци (слика 6.11.).



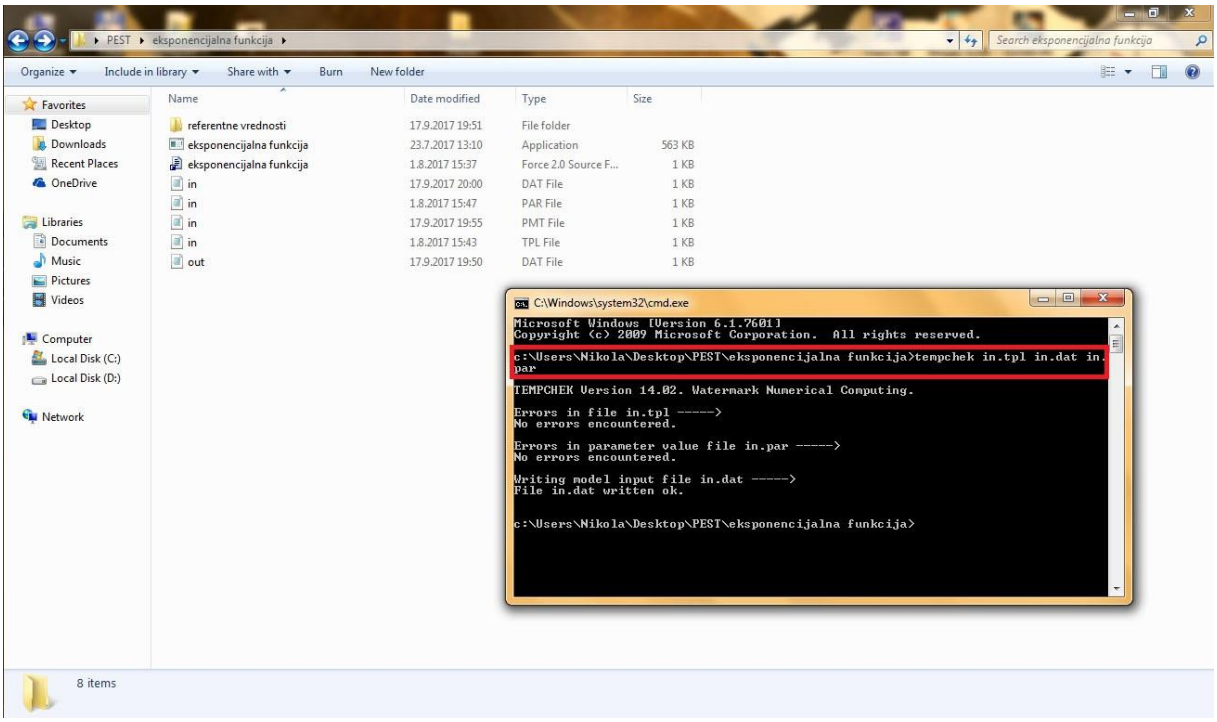

*Слика 6.11. Структура .pmt датотеке*




*Слика 6.12. Структура датотеке in.par*

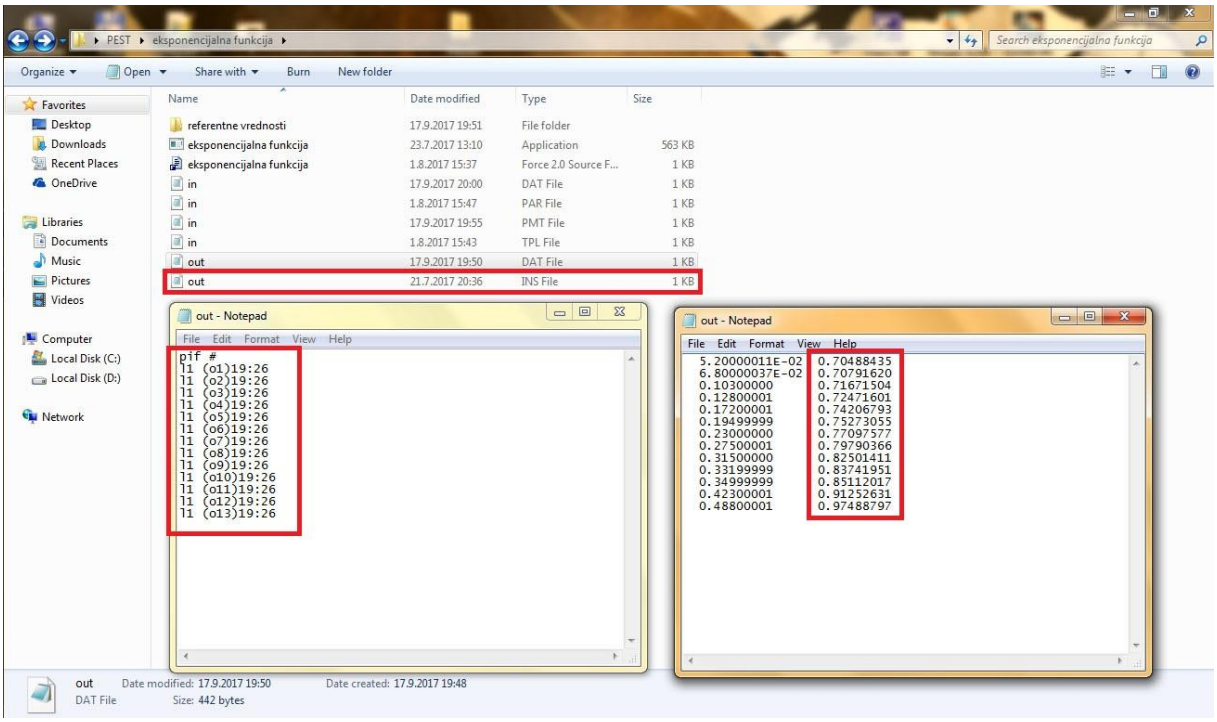
Следећа датотека коју је потребно креирати је in.par. У овој датотеци дефинише се вредност од које ће PEST почети варирање датог параметра. У овом случају задате су исте вредности које су коришћене у улазној датотеци када је програм извршен у директоријуму експоненцијална функција (слика 6.12.).

Када су све наведене датотеке креиране, потребно је проверити коректност истих. Потребно је покренути command prompt у терминалу где су датотеке смештене и унети команду tempchek in.tpl in.dat in.par (слика 6.13.). Након овога, у случају да је дошло до грешке приликом креирања неке од датотека, програм ће јавити грешку, и у којој датотеци се налази грешка. Ако је све коректно креирано може се наставити даље.



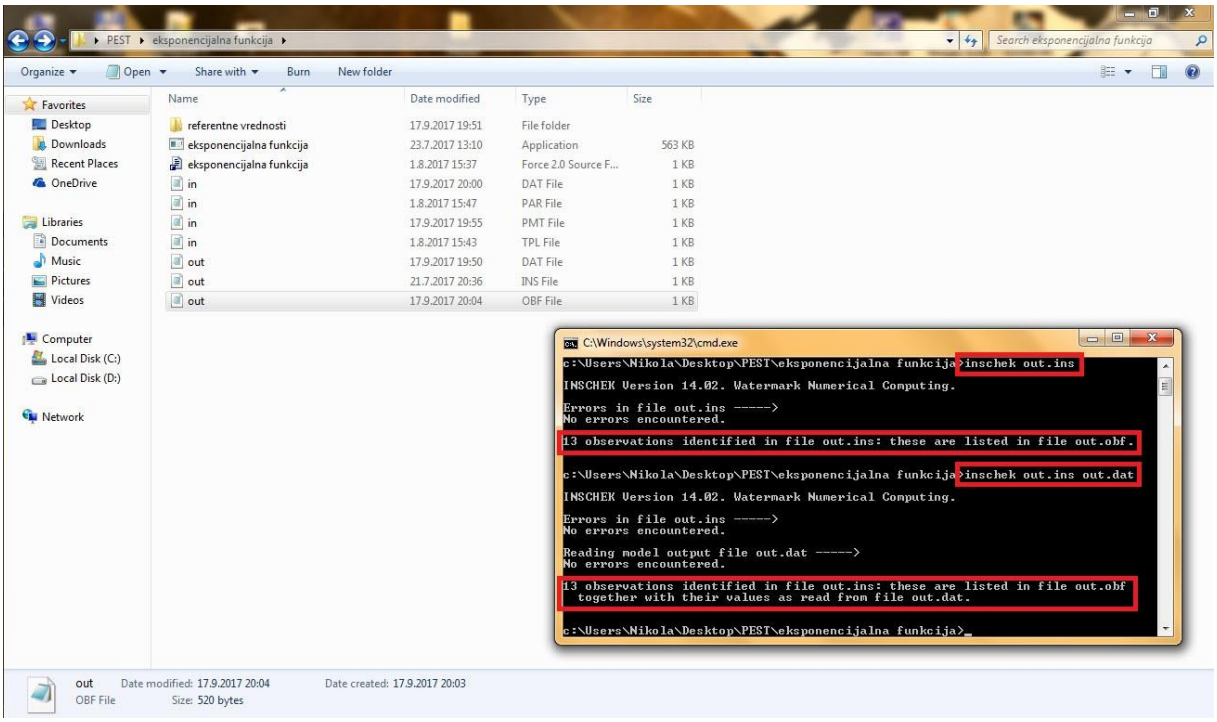

*Слика 6.13. Провера коректности претходно креираних датотека*

Следећи сет датотека који је потребно креирати су инструкционе датотеке. Прво је потребно креирати PEST Instruction File. У овој датотеци потребно је дати инструкцију PEST-у како да маркира вредности излаза из излазне датотеке out.dat. У првој линији кода датотеке out.ins потребно је унети pif #. pif је скраћено од PEST instruction file a # је симбол којим ће бити маркирани карактеристични редови у излазној датотеци помоћу којих ће се одредити из ког реда тачно креће учитавање вредности излазних параметара. Симболима o1, o2, o3..., означавају се редни бројеви вредности излазних параметара које PEST треба да учита из излазне датотеке. Након симбола потребно је унети број поља где је смештена вредност, тј. колико поља заузима вредност која треба да буде учитана (слика 6.14.).



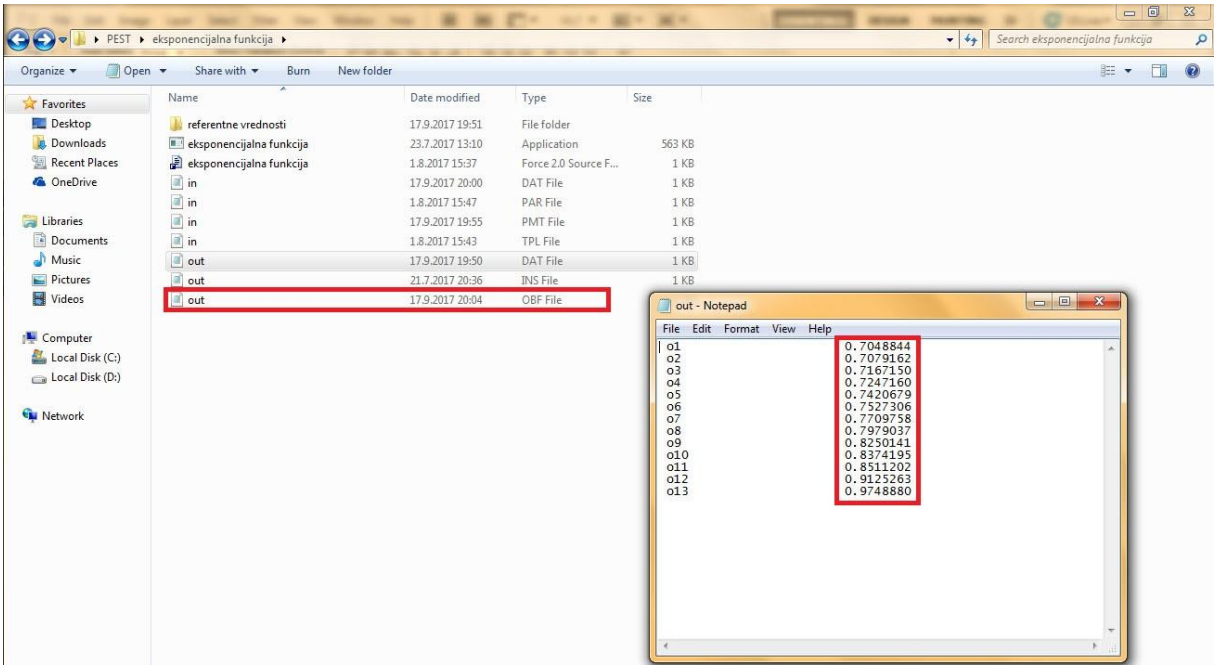

*Слика 6.14. Изглед датотеке out.ins*

Након креирања ове датотеке, потребно је покренути command prompt, и издати команду inschek out.ins. На овај начин, аутоматски ће се креирати датотека out.obf у којој ће редом бити излистани симболи којима се означавају редни бројеви вредности које требају да буду учитане из излазне датотеке. Затим, у command prompt-у уносимо инструкцију inschek out.obf out.dat (слика 6.15.).



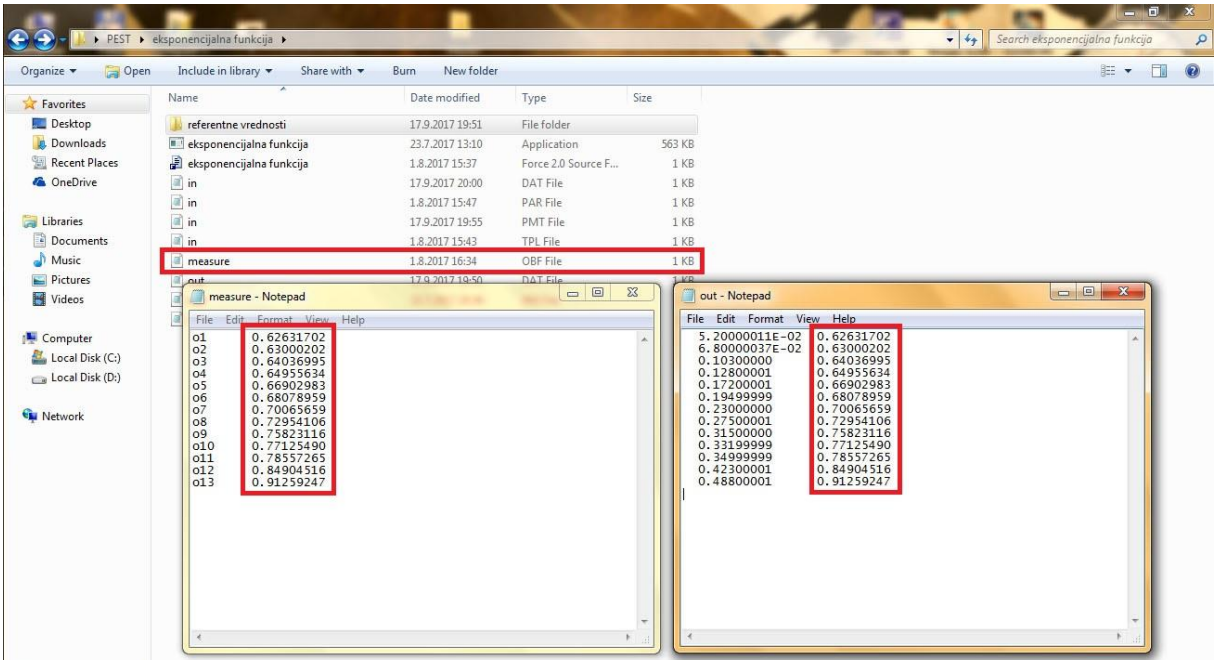

*Слика 6.15. Креирање датотеке out.obf*

На овај начин, у датотеци out.obf, биће излистане вредности параметара из излазне датотеке као што је приказано на слици 6.16.



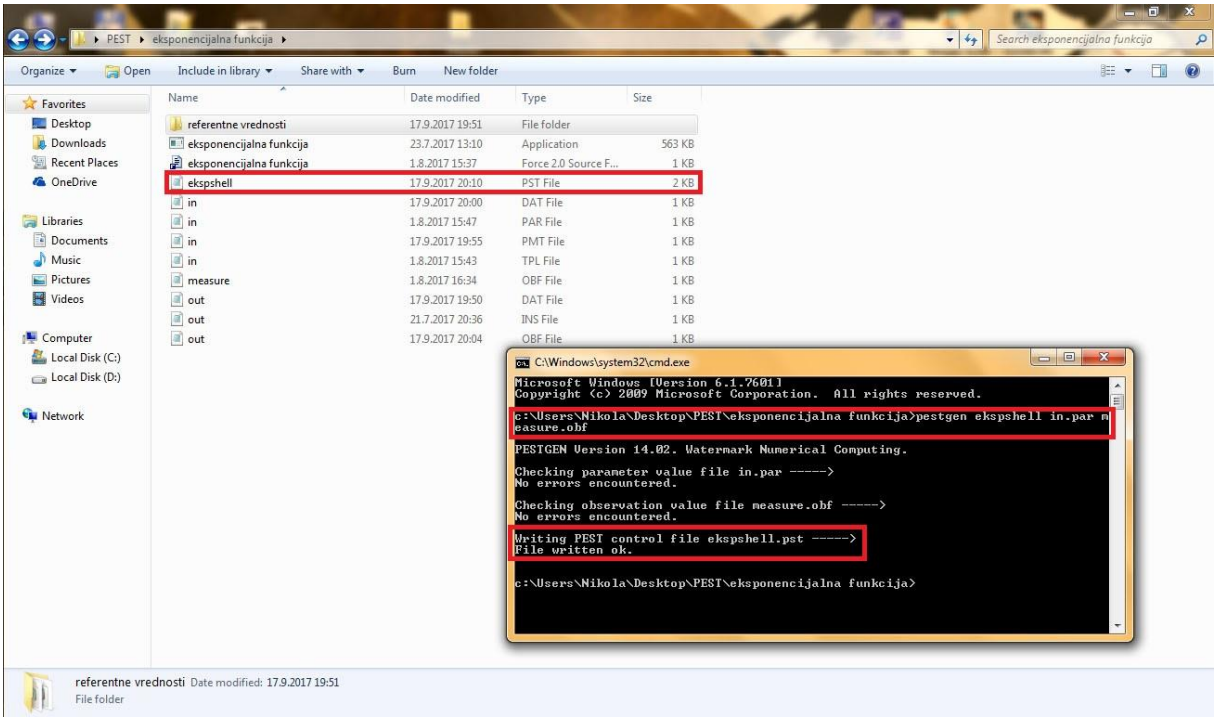

*Слика 6.16. Структура датотеке out.obf*

Следећи корак је креирање measure.obf датотеке, која служи да PEST-у обезбеди вредности са којима ће поредити резултате прорачуна при свакој новој итерацији. У овом случају за вредности које су унешене у measure.obf, узимају се вредности излазних параметара из прорачуна који је извршен у директоријуму референтне вредности као што је приказано на слици 6.17.




*Слика 6.17. Структура датотеке measure.obf*

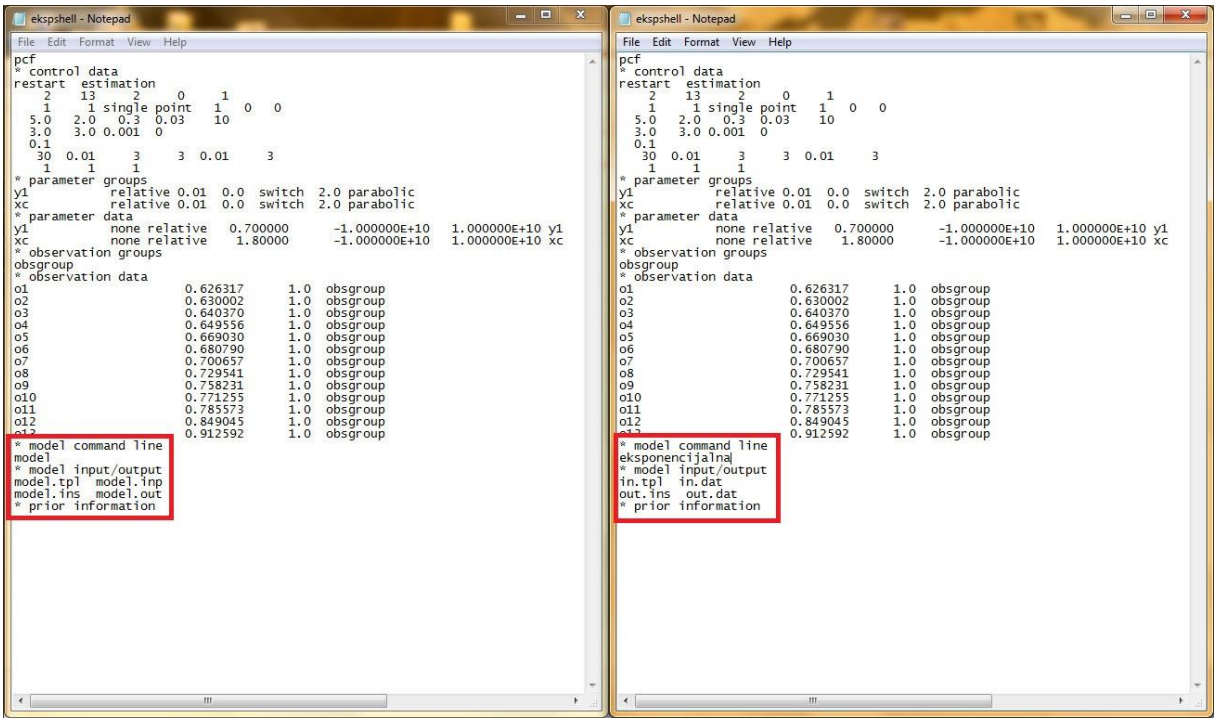
Након што је креирана датотека measure.obf, потребно је генерисати финалну датотеку која ће објединити све инструкције и параметре из свих претходно креираних датотека. То се ради покретањем command prompt-a у терминалу где су смештене све претходно креиране датотеке, и издавањем команде pestgen ekspshell in.par measure.obf. ekspshell је име датотеке коју ће PEST генерисати у овом случају. Овако креирана датотека има екстензију .pst (слика 6.18.).




*Слика 6.18. Генерисање датотеке ekspshell.par*

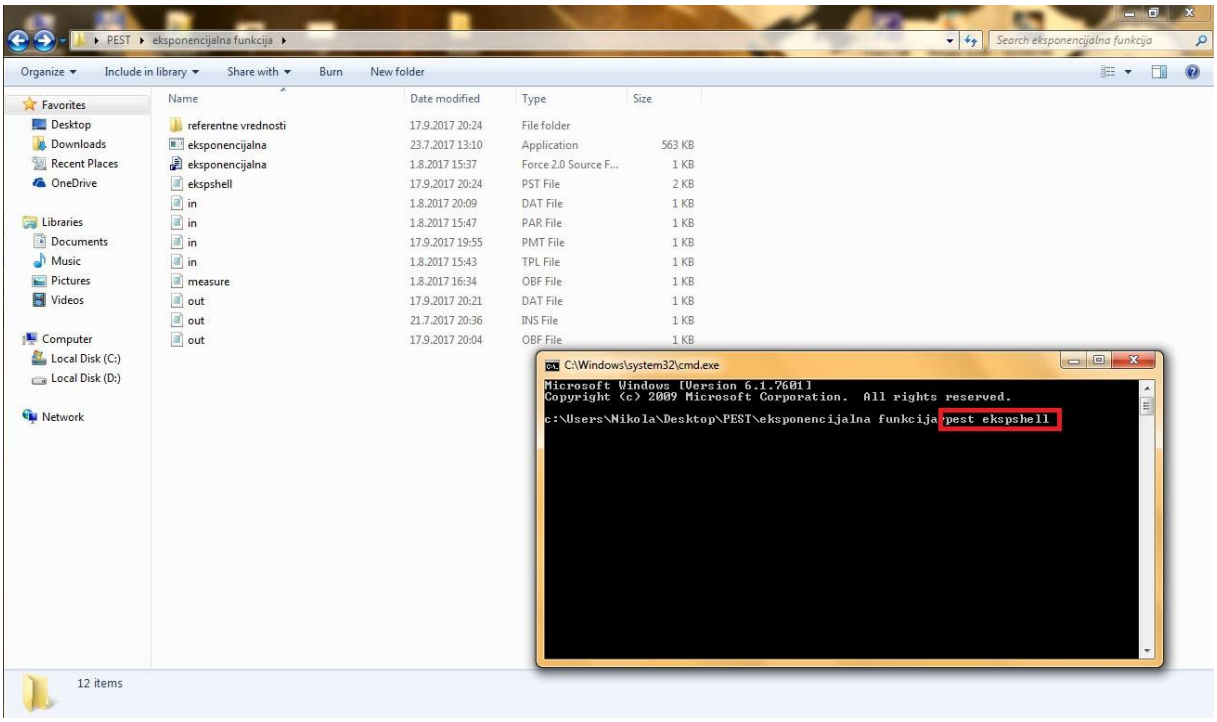
Пре него што се покрене прорачун, потребно је извршити мале корекције у оквиру ове датотеке. Као на слици, потребно је унети име exe-а који се извршава. У овом случају то је eksponencijalna funkcija, с тим што програм не може да учита назив са резмаком па је име кориговано само на eksponencijalna. Такође, то је потребно урадити и са програмом у директоријуму експоненцијална функција.

Следеће је потребно унети имена template датотеке и одговарајуће улазне датотеке, а одмах испод имена instruction датотеке као и одговарајуће излазне датотеке. Овим је и потпуно завршено са креирањем ekspshell датотеке (слика 6.19.).




*Слика 6.19. Корекција датотеке ekspshell.par*

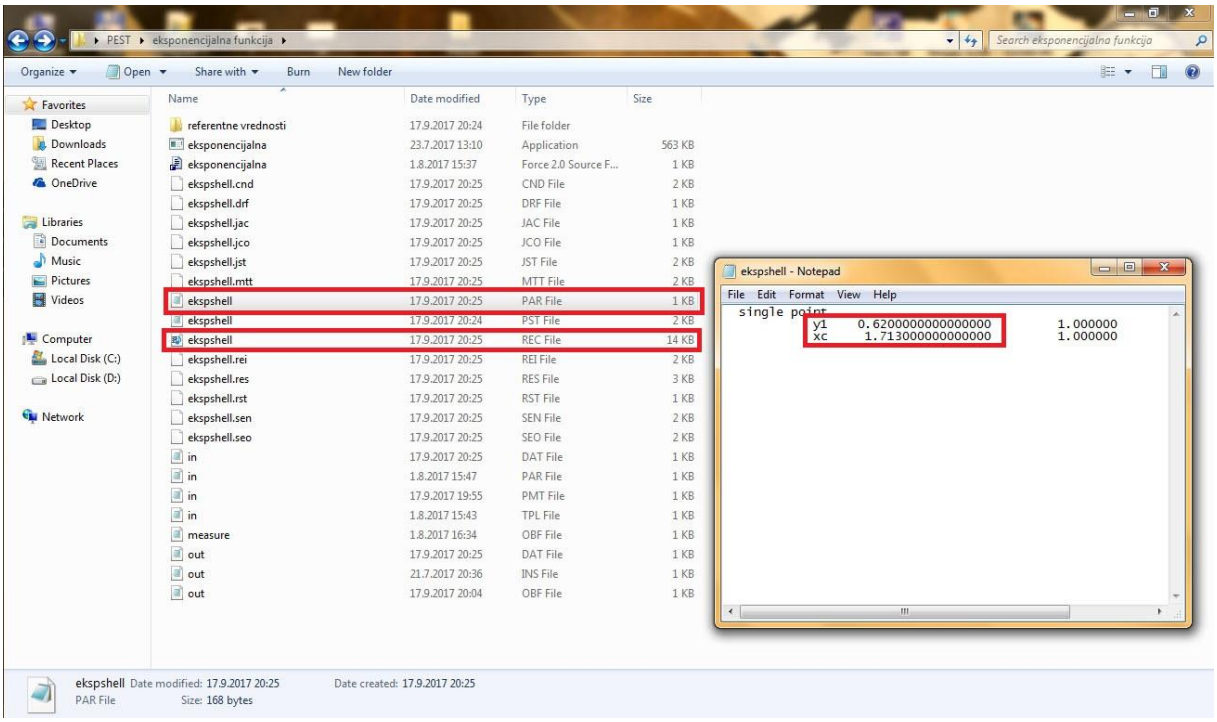
Сада је све спремно за покретање саме оптимизације. Потребно је покренути command prompt у директоријуму експоненцијала функција и унети наредбу pest ekspshell (слика 6.20.). На овај начин покреће се оптимизација.




*Слика 6.20. Издавање команде за покретање PEST оптимизације*

Након извршеног прорачуна, PEST генерише ekspshell.par датотеку у којој су смештени резултати оптимизације (слика 6.21.). На слици се може видети да у овом случају не постоји ни најмање одступање тражених параметара. То се може приписати релативно једноставном матемеатичком функцијом која се извршава у програму eksponencijalna funkcija, и тиме што су вредности које су задате, а од којих креће оптимизација релативно близу жељеном резултату.

ekspshell.rec претставља датотеку у коју PEST записује цео поступак прорачуна оптимизације. Овде се може наћи колико је итерација извршено, које вредности су додељиване траженим параметрима приликом сваке итерације и др.

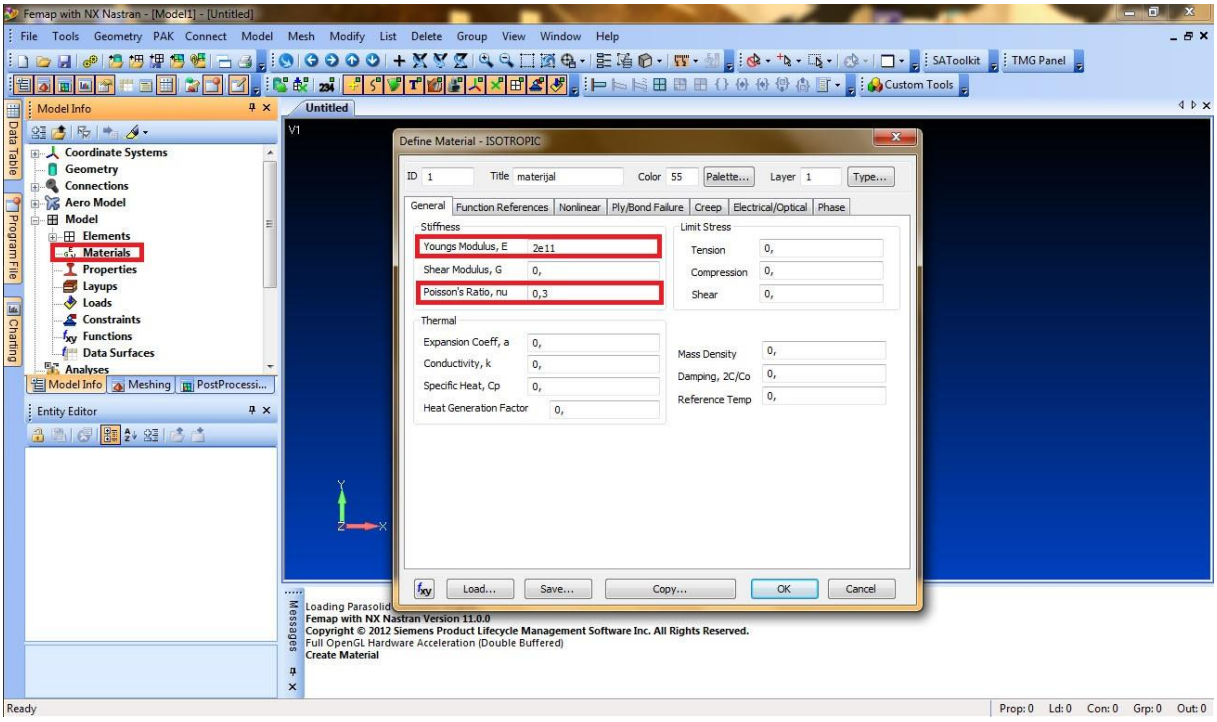



*Слика 6.21. Приказ добијених параметара у датотеци ekspshell.par*

## 6.3. Креирање FEM модела у програмском пакету FEMAP

Модел који се креира у сврху излагања методологије оптимизације треба бити што једноставнији ради лакшег праћења и разумевања поступка. У овом случају модел ће имати најједноставнију могућу геометрију (коцка), која ће претстављати један коначни елемент са осам чворова. Модел је потребно ограничити тако да оптерећења која делују на модел дају реалну слику деформација и напона у материјалу. Ово је јако битно јер ће се у оптимизацији тражити модул еластичности у зависности од померања чворова коначног елемента под дејством сила у чворовима. Даље ће бити изложен поступак креирања модела у програмском пакету FEMAP.

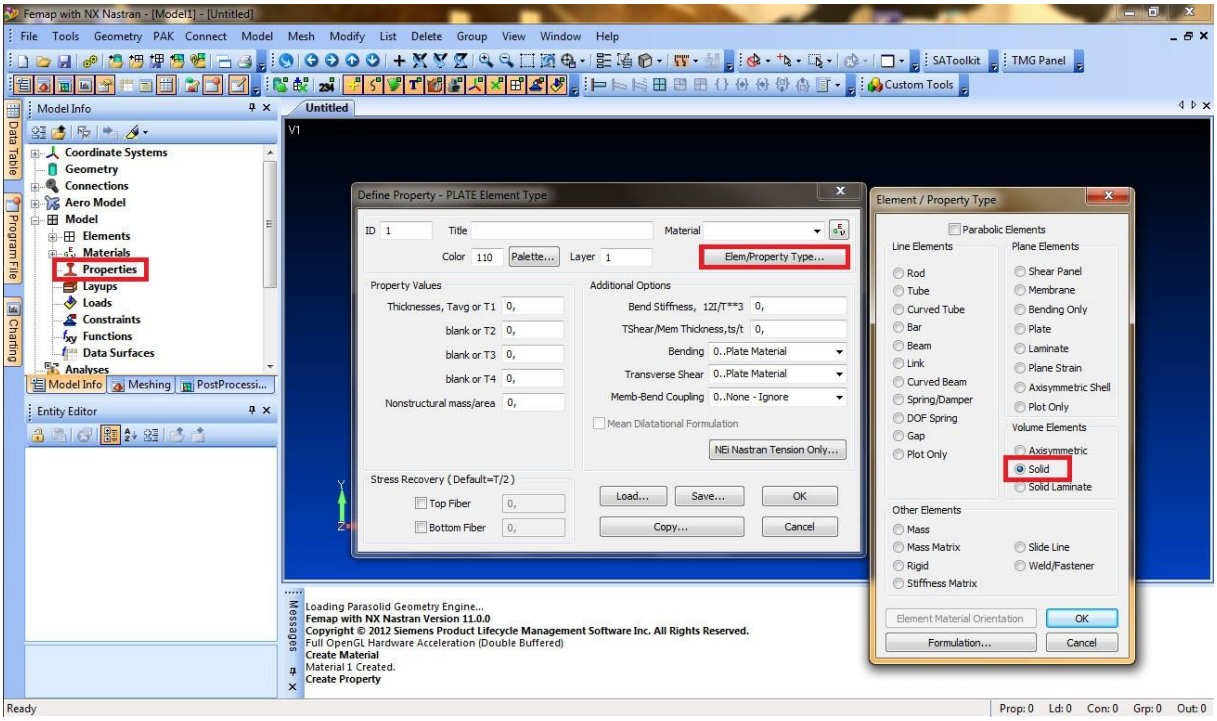
Потребно је дефинисати карактеристике материјала који ће касније бити додељен моделу. Са леве стране екрана, под картицом Model Info налази се стабло модела. Потребно је десним кликом изабрати грану Materials и изабрати креирање новог материјала. Након што је то урађено, отвара се прозор Define Material и у њему је потребно дефинисати име материјала (у овом случају име је материјал), и карактеристике материјала. У овом примеру су дефинисани Јангов модул еластичности и Поасонов коефицијент, $\mathrm{E} = 2\mathrm{E}11$ и $nu = 0.3$. Овим је завршено дефинисање материјалних карактеристика (слика 6.22.).




*Слика 6.22. Дефинисање материјалних карактеристика модела*

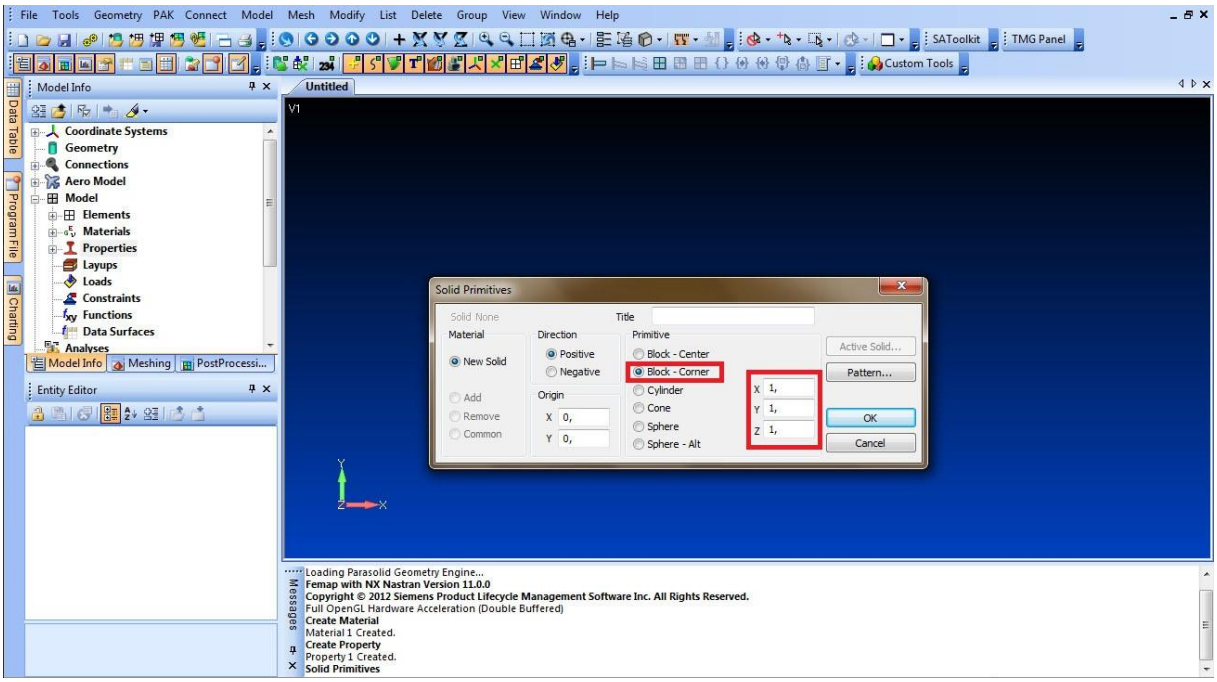
Следећи корак је дефинисање property-а материјала. У стаблу модела потребно је десним кликом изабрати грану Properties и изабрати креирање новог property-а. Након што је то урађено отвара се нови прозор Define Property у коме треба изабрати тастер Element Property Type. У прозору који се отвори потребно је чекирати Solid. Након тога потребно је дати име property-ју (у конкретном случају solid) и у падајућем менију

доделити му материјал који је креиран у претходном кораку. На овај начин је property дефинисан (слика 6.23.).



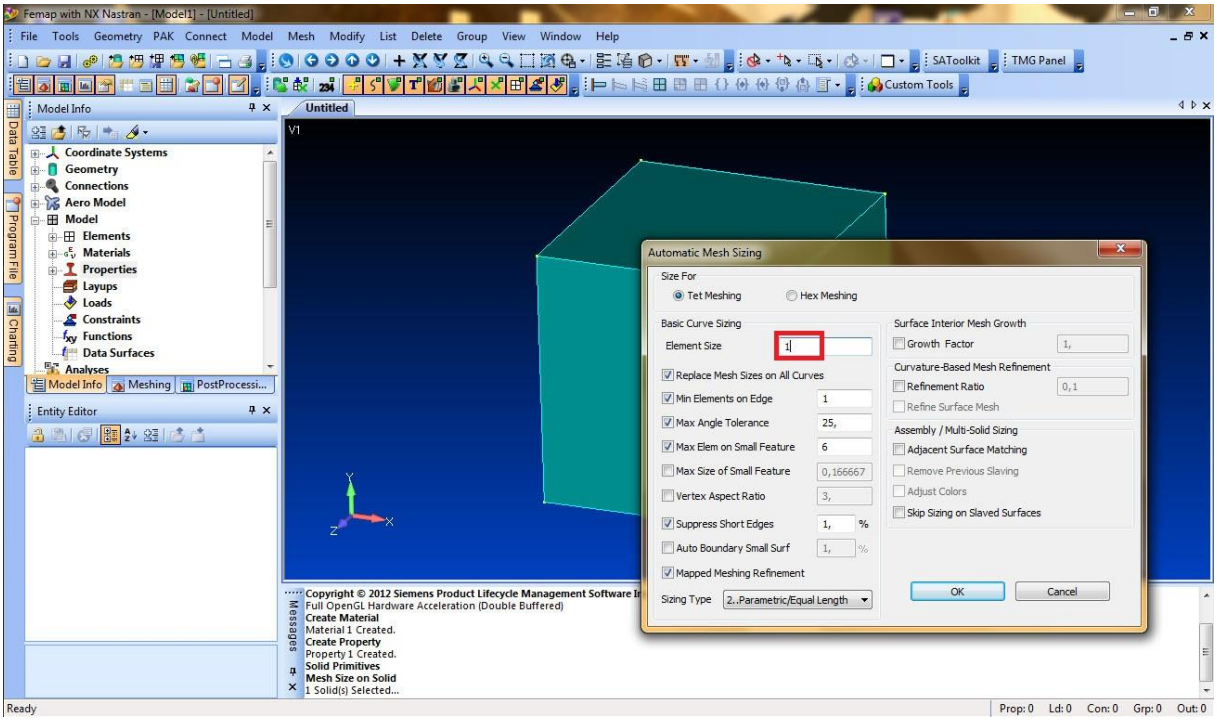

*Слика 6.23. Дефинисање property-ја модела*

Након тога, потребно је креирати геометрију модела. Геометрија модела је коцка која ће претстављати један коначни елемент. Потребно је изабрати Gemoetry > Solid > Primitives, и у прозору који се отвори чекирати Block – Corner, као што је приказано на слици 6.24. Овим је креирана геометрија модела.

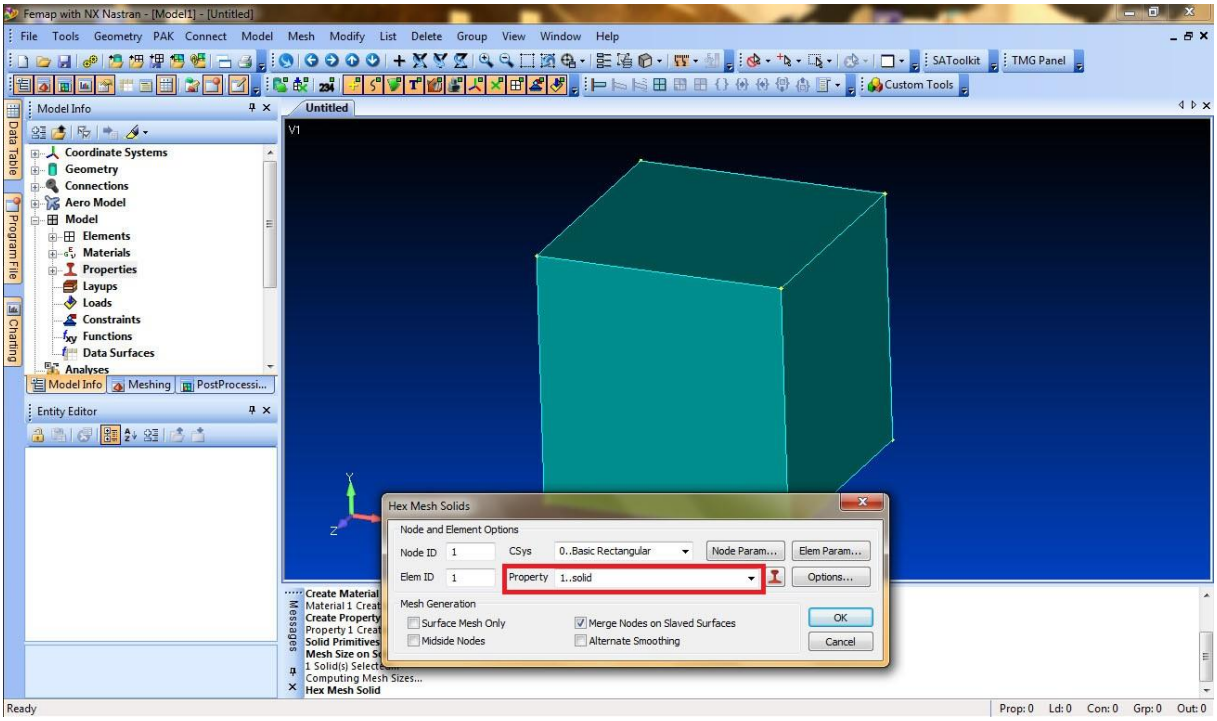



*Слика 6.24. Креирање геометрије модела*

Следеће што је потребно је дефинисати мрежу модела. Потребно је изабрати Mesh > Mesh Control > Size on Solid. У прозору који се отвори потребно је дефинисати Element Size што ће у овом случају бити 1 како би креирана геометрија претстављала један коначни елемент (слика 6.25.).



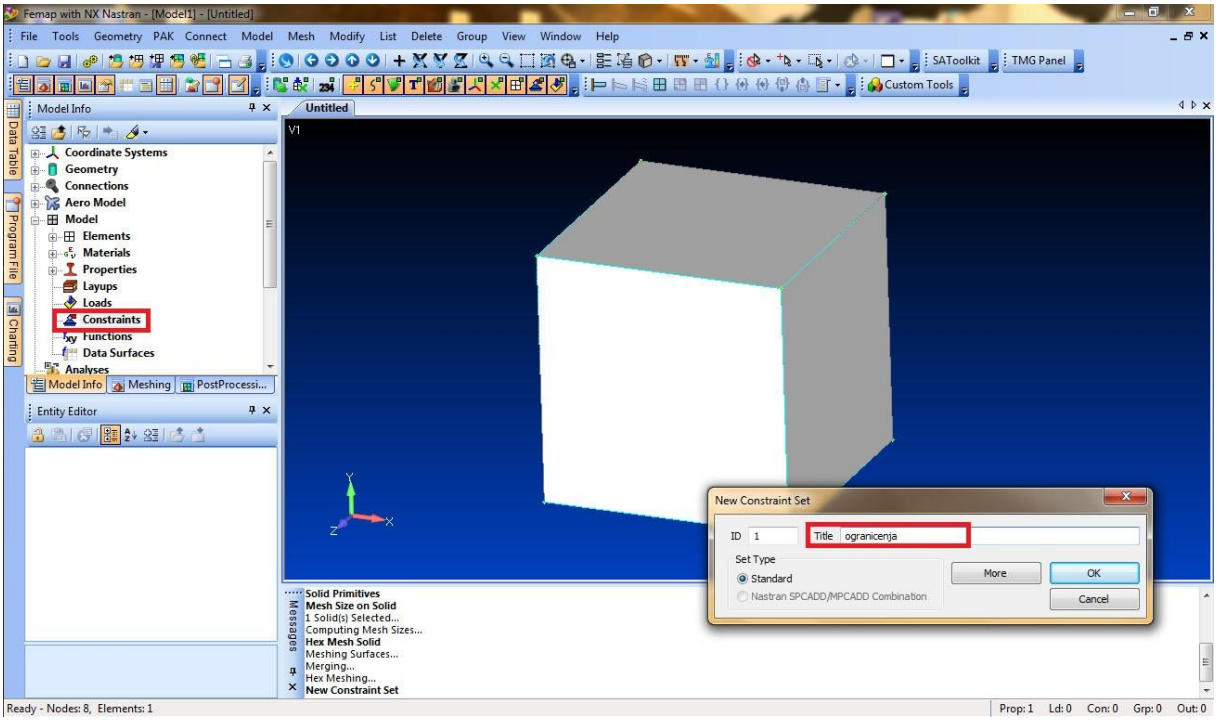

*Слика 6.25. Креирање мреже модела*




*Слика 6.26. Додељивање мреже моделу*

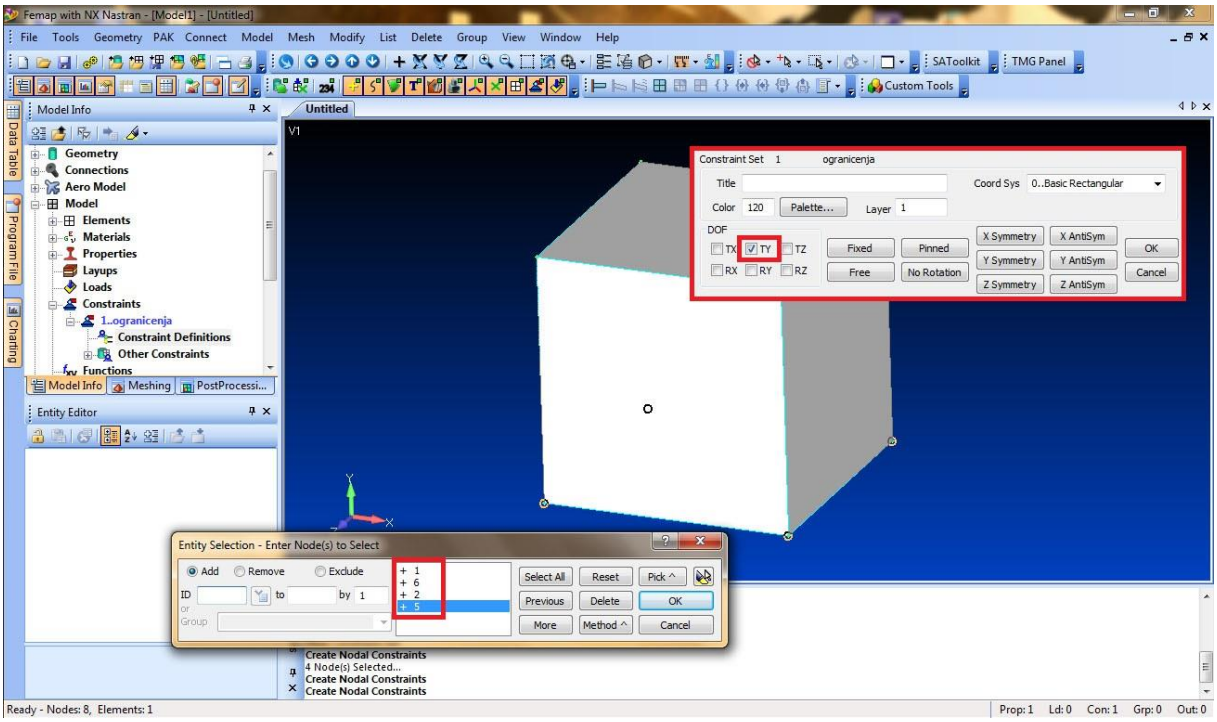
Након дефинисања мреже потребно је мрежу доделити геометрији. Потребно је изабрати Mesh > Geometry > HexMesh Solids и у падајућем менију изабрати претходно креиран property. На овај начин је мрежа додељена моделу (слика 6.26.).

Након овога потребно је задати ограничења моделу. У стаблу модела потребно је десним кликом миша изабрати Constraints и дефинисати име за New Constraint Set (у овом случају названим ограничења) (слика 6.27.).



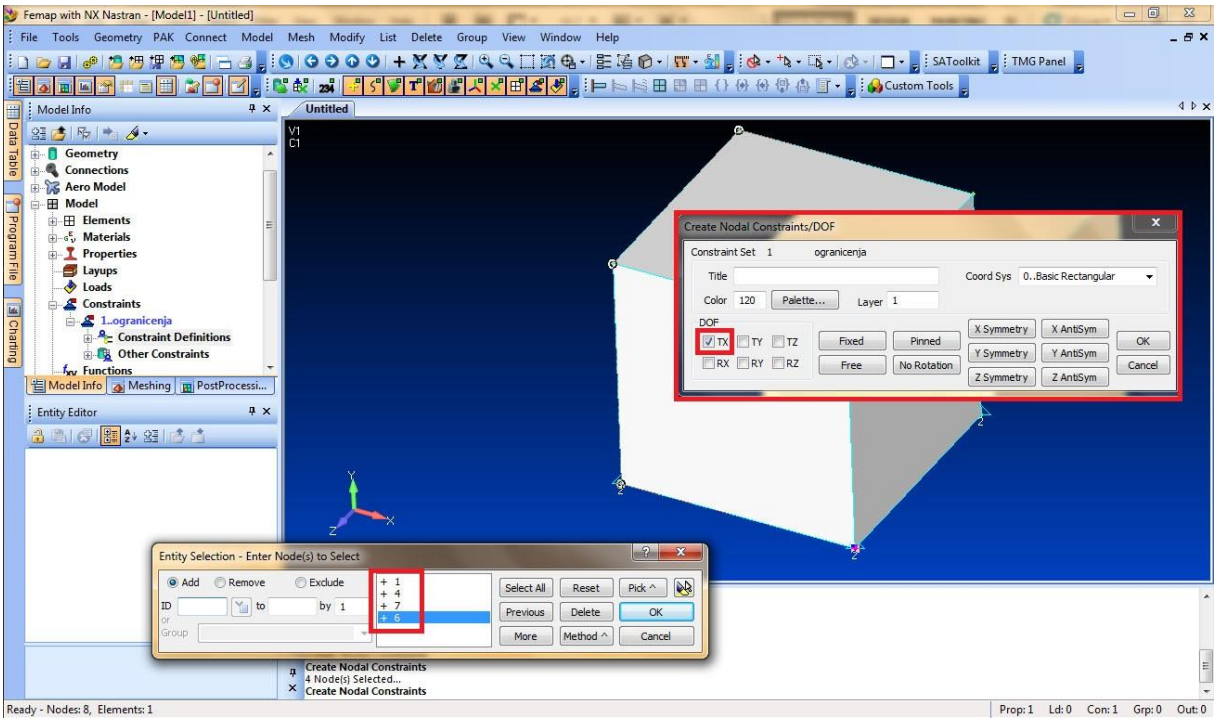

*Слика 6.27. Креирање сета ограничења*

Када је constraints set креиран, потребно је у стаблу модела десним кликом миша изабрати Constraint Definitions и изабрати Nodal. Након тога потребно је селектовати чворове које требају бити ограничени и изабрати да ли су ограничене транслација или ротација и по којој оси. У овом примеру су ограничени чворови у *xz* равни а ограничена им је транслација по *y* оси, као на слици 6.28.



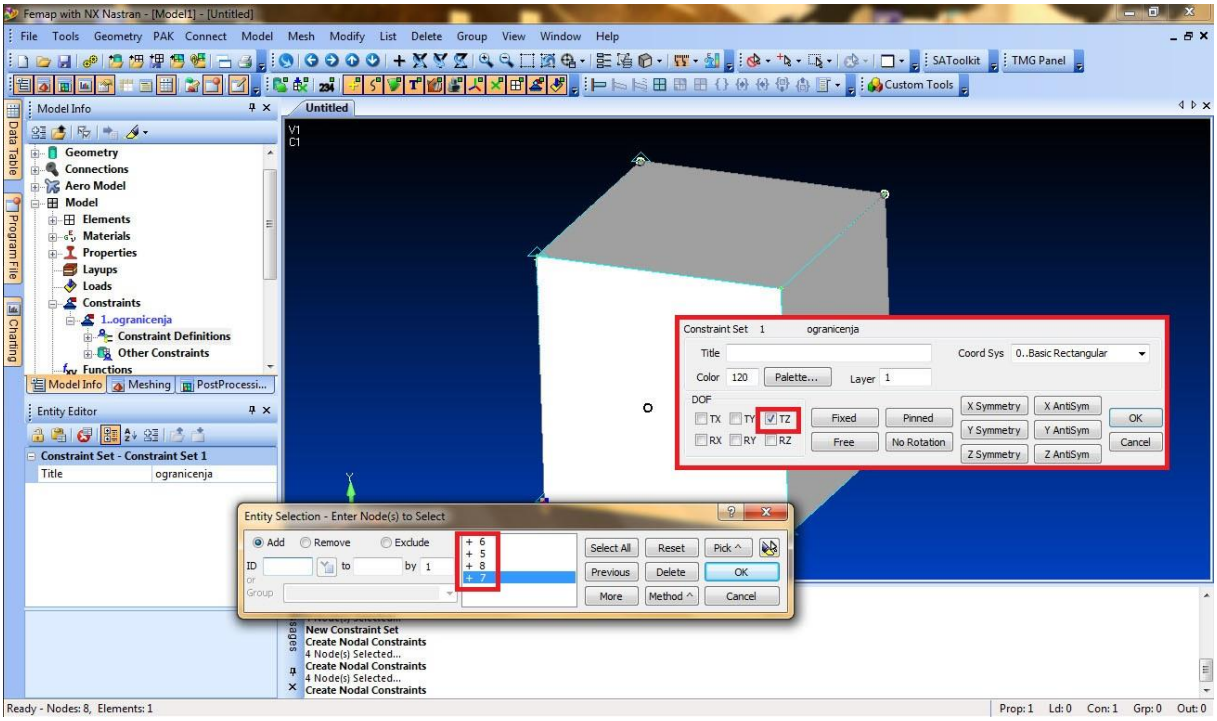

*Слика 6.28. Ограничавање чворова по y оси*

Идентичан поступак је потребно поновити за сва остала ограничења које је потребно доделити моделу. На слици 6.29. приказано је чворовима у *yz* равни оганичена транслација по *x* оси.

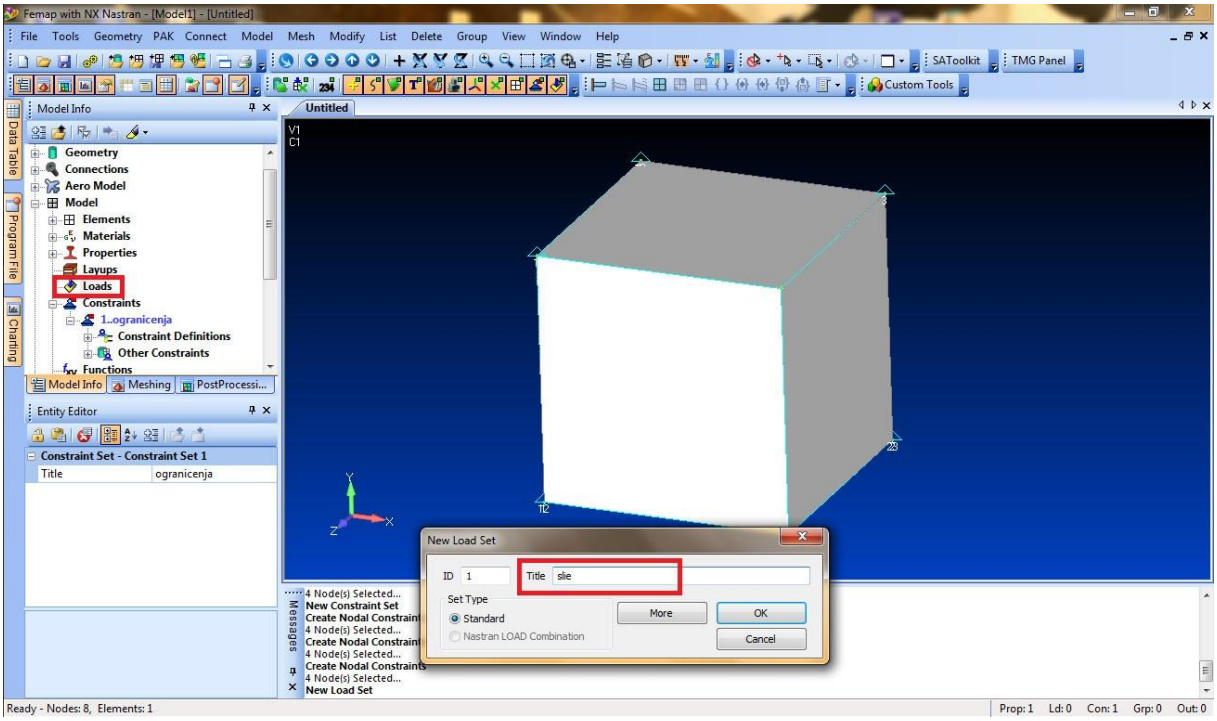



*Слика 6.29. Ограничавање чворова по x оси*

Идентичан поступак је поновљен при ограничавању чворова који се налазе у *xy* равни а ограничена им је транслација по *z* оси као што је приказано на слици 6.30.



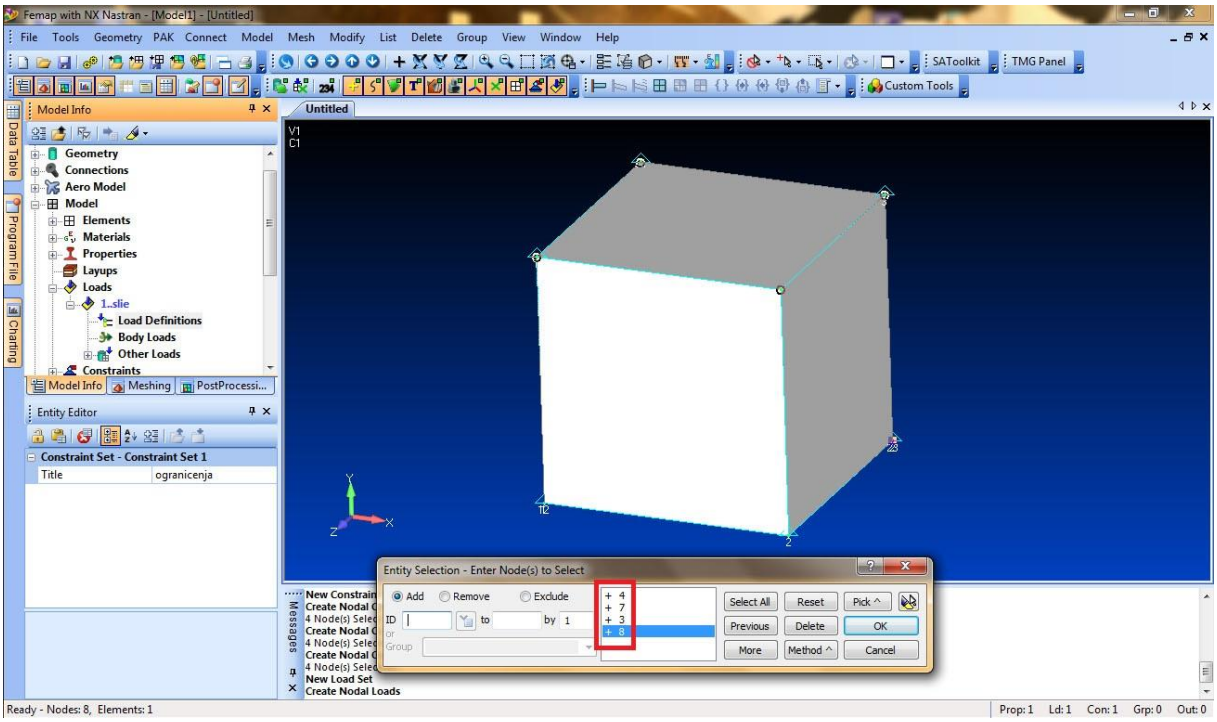

*Слика 6.30. Ограничавање чворова по z оси*




*Слика 6.31. Креирање сета оптерећења*

Након што су дефинисана и задата сва ограничења, потребно је моделу задати оптерећења. У стаблу модела потребно је десним кликом миша изабрати Loads и дефинисати име за New Load Set (у овом случају названим силе). Овим је извршена припрема за доделу ограничења моделу (слика 6.31.).

У грани модела потребно је десним кликом одабрати Load Definitions, а затим у новоотвореном прозору изабрати чворове које треба оптеретити. У овом случају то су чворови којима није ограничена транслација по *y* оси (слика 6.32.).

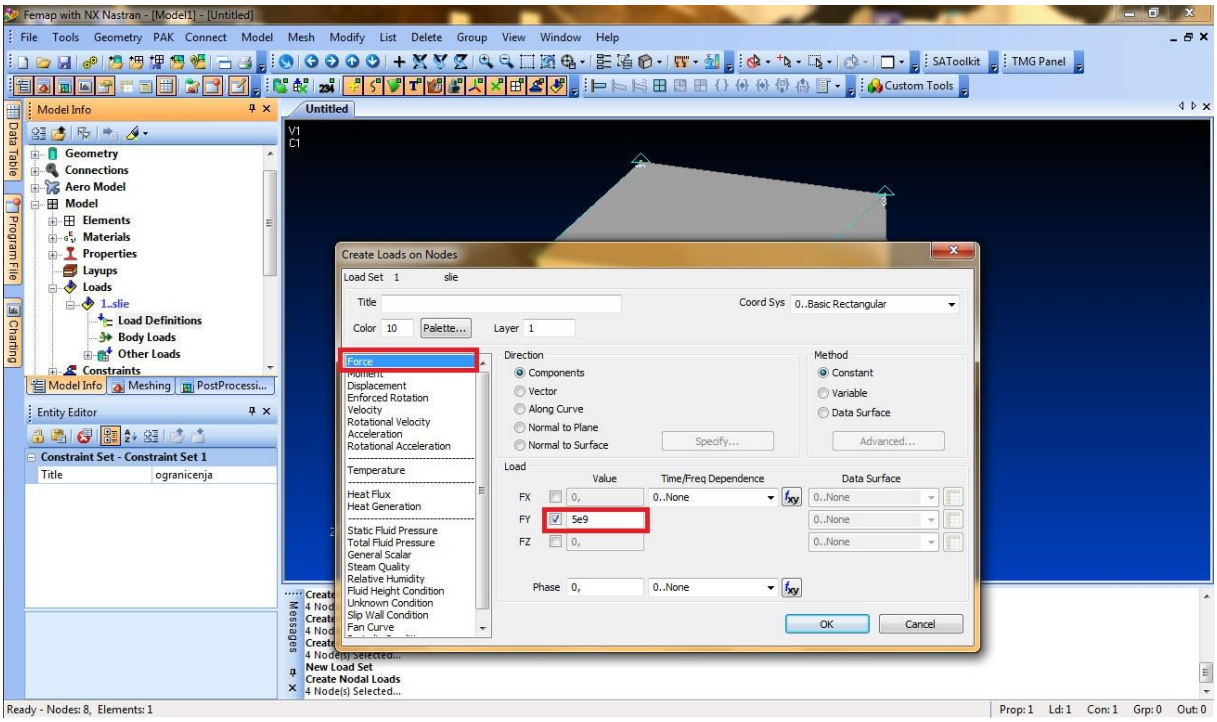



*Слика 6.32. Маркирање чворова у којима ће деловати силе*

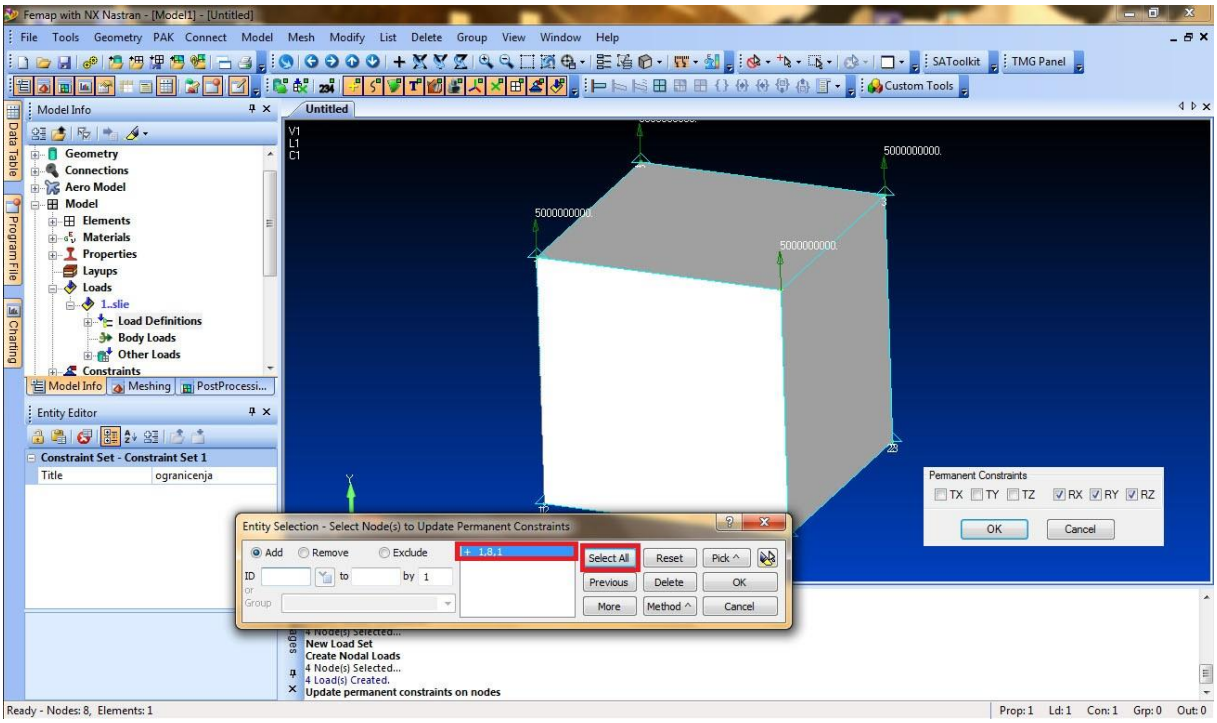
Када се изврши потврда изабраних чворова, у прозору који се отвори, потребно је изабрати врсту оптерећења, начин на који делује и интензитет. У датом примеру у чворовима делују силе у супротном смеру од *y* осе (слика 6.33.).

На крају је потребно задати глобална ограничења моделу, која делују на све чворове модела. Треба изабрати Model > Update Other > Permanent Constraint, а затим у прозору који се отвори изабрати све чворове. Њима је потребно ограничити ротацију по све три осе. На овај начин је готова припрема модела и може се приступити процесирању (слика 6.34.).

Модел је пре покретања анализе потребно сачувати на жељеној локацији (у овом примеру модел је сачуван на desktop-у под именом коцка)

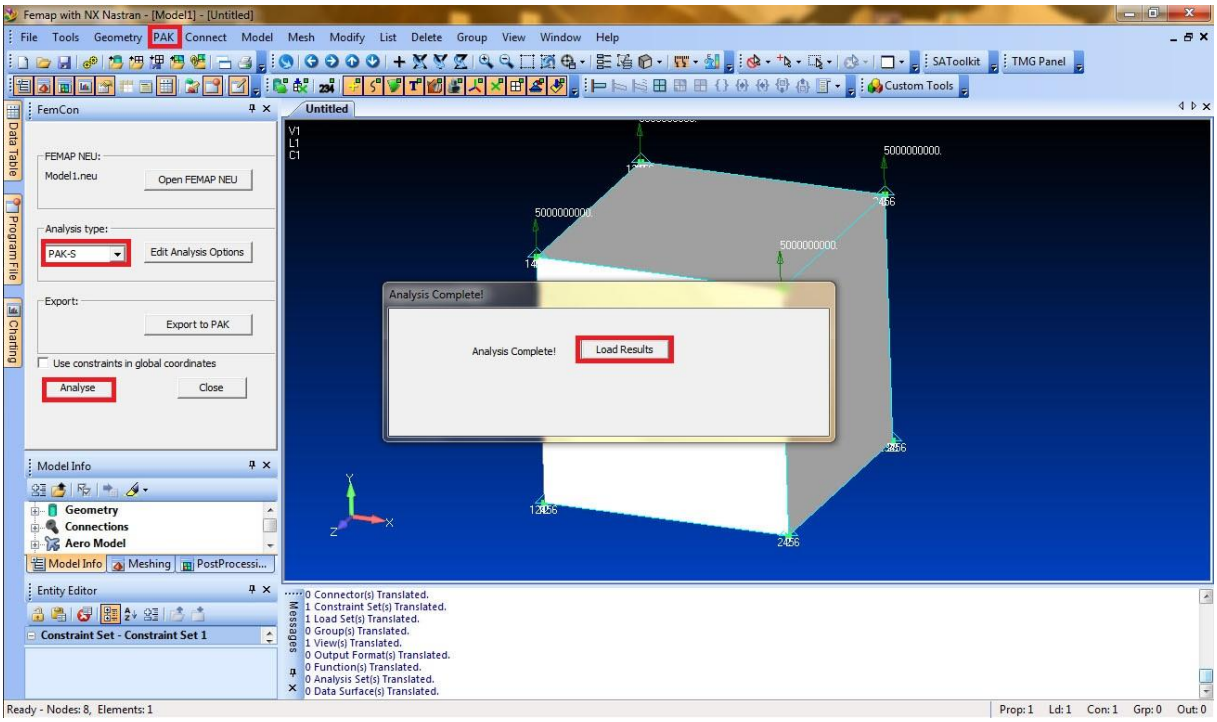



*Слика 6.33. Одабир врсте, правца и интензитета оптерећења*




*Слика 6.34. Дефинисање глобалних ограничења*

Процесирање примера који се обрађује вршено је програмским додатком PAK. Потребно је изабрати PAK, а затим изабрати у падајућем менију врсту анализе. У овом случају рађена је структурна анализа, и због тога је изабран PAKS. Након што се то изабере потребно је изабрати тастер Analyse (слика 6.35.).



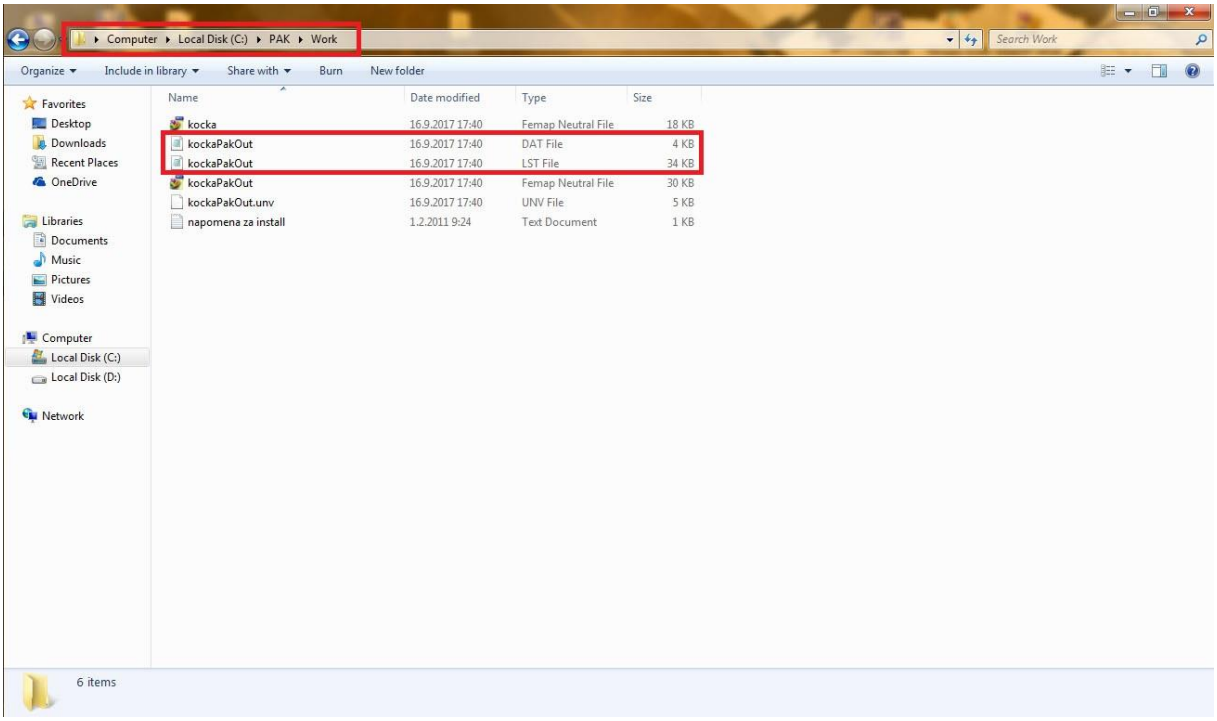

*Слика 6.35. Покретање анализе програмским додатком PAK-S*

Потребно је сачекати неколико тренутака да се анализа изврши, а затим у прозору који се отвори потребно је изабрати Load Results, како би резултати били учитани у програм. Овим је завршена припрема модела и резултата који требају ући у процес оптимизације који је приказан у наставку [12].

## 6.4. Идентификација параметара материјалних карактеристика модела

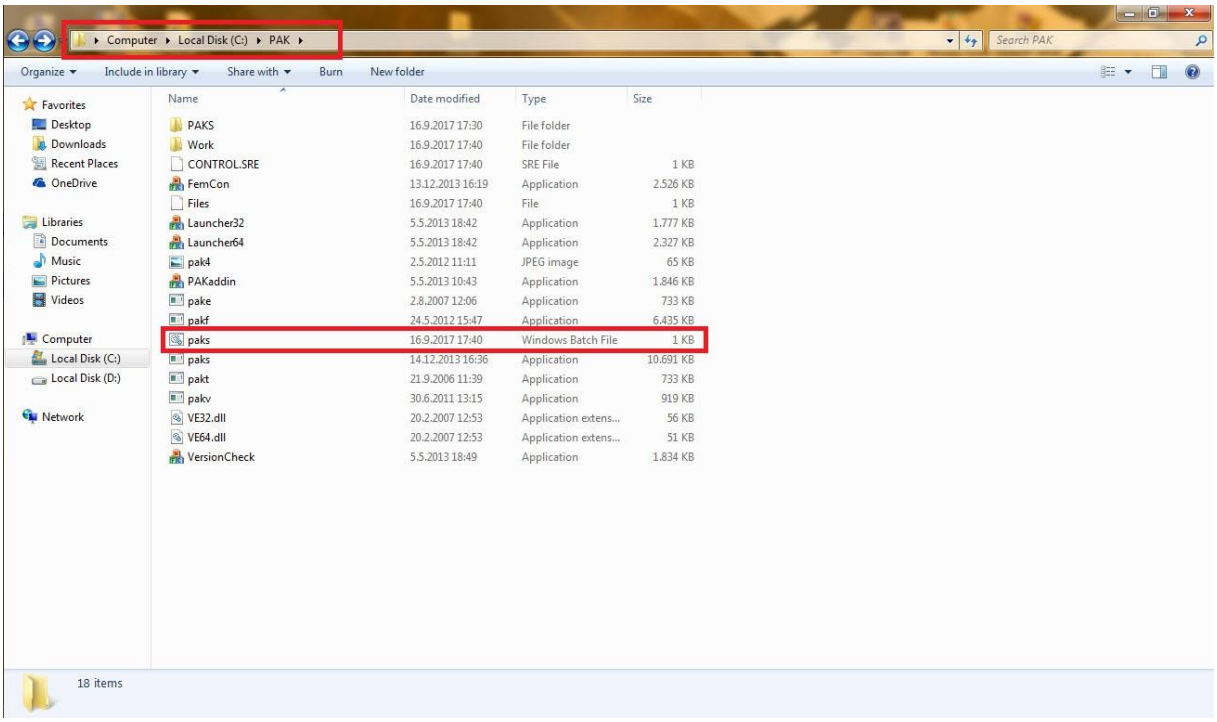
Циљ оптимизације која ће бити рађена у овом примеру јесте да се за циљане излазне вредности одреди једна од материјалних карактеристика модела. Идеја је да се са промењеним Јанговим модулом еластичности добију другачија померања чворова од оних која су добијена у предходно рађеној анализи модела, и користећи такве вредности померања као референтне резултате, процесом оптимизације нађе модул еластичности са којим су добијене такве вредности померања.

Потребно је припремити све потребне датотеке (улазна датотека прорачуна модела, као и exe који извршава тај прорачун) које су потребне за поступак оптимизације. На локацији Local Disk (C) > PAK > Work, налазе се улазна и излазна датотека прорачуна који је извршен са моделом kocka. Улазна датотека има аутоматски генерисан назив kockaPakOut.dat, а излазна kockaPakOut.lst. (слика 6.36.).



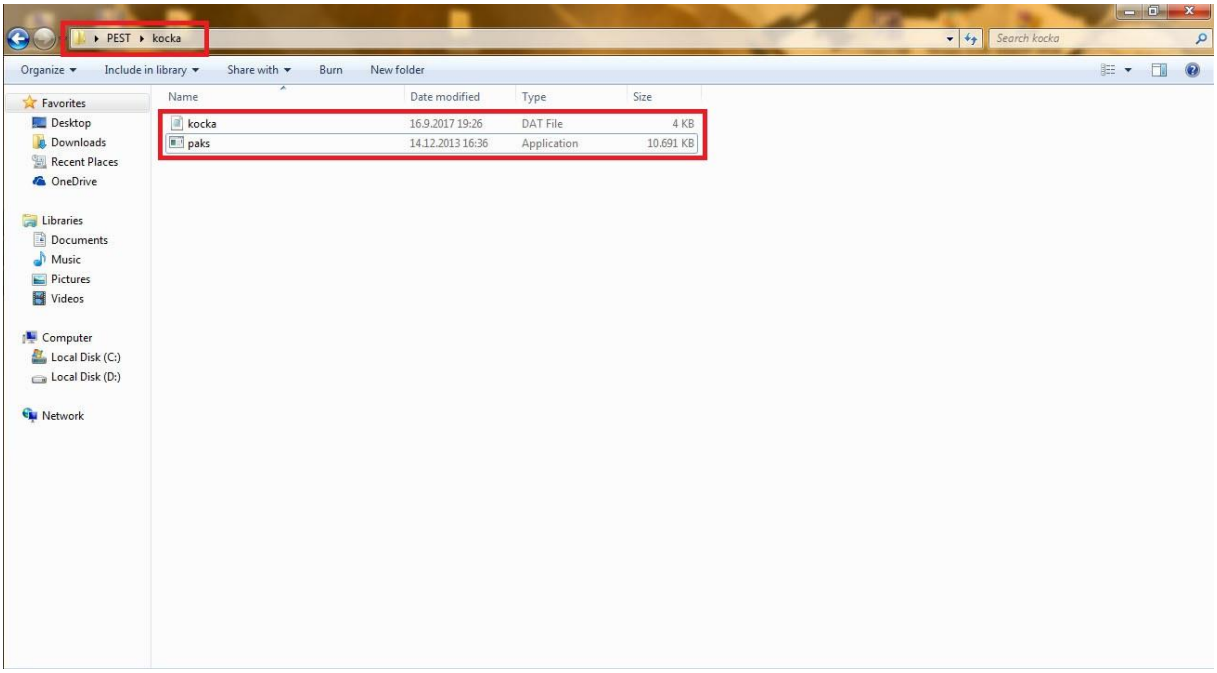

*Слика 6.36. Улазна и излазна датотека прорачуна модела у терминалу PAK-а*

На локацији Local Disk (C) > PAK, налазе се програми који извршавају прорачун. Пошто је на моделу kocka вршена структурна анализа она је извршена апликацијом paks (слика 6.37.).



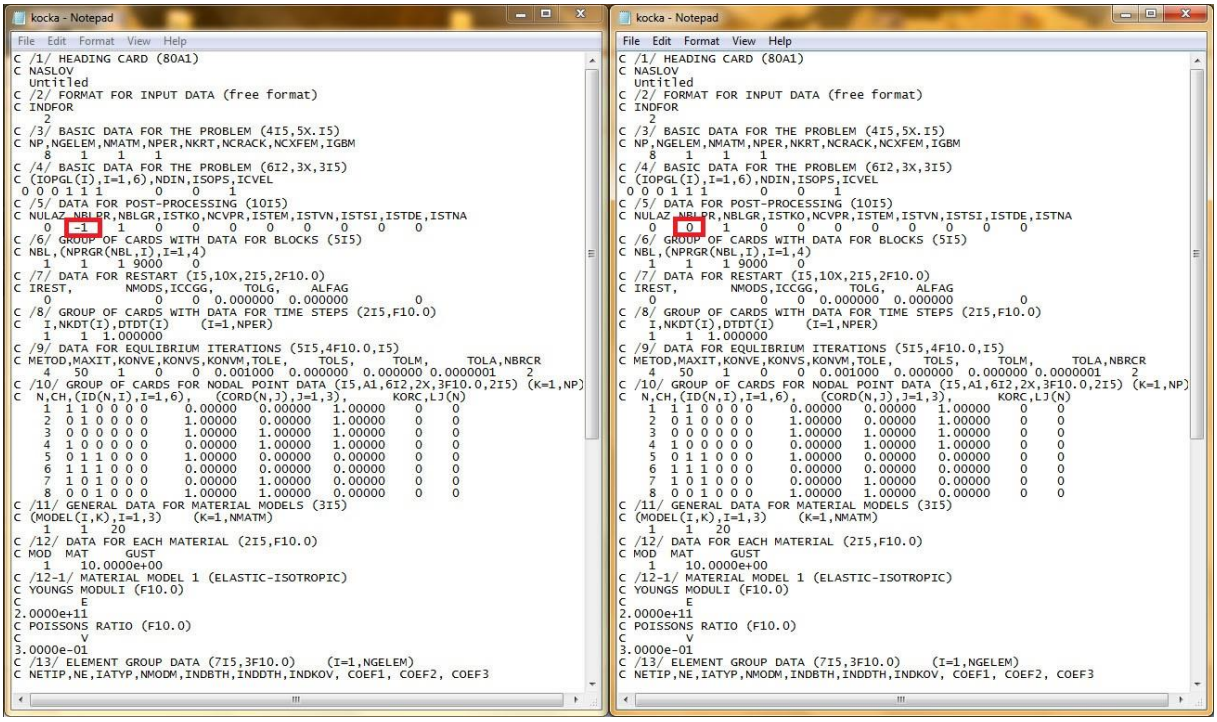

*Слика 6.37. paks у инсталационом терминалу PAK-а*

Потребно је креирати директоријум у ком ће бити извршена оптимизација, а услов је да буде смештен у директоријуму PEST, како би PEST-ови извршни програми могли да изврше оптимизацију. За потребе овог примера у директоријуму PEST, креиран је директоријум kocka, у који су копирани улазна датотека прорачуна kockaPakOut, као и апликација која извршава прорачун, paks. Ради лакше манипулације улазна датотека прорачуна, kockaPakOut, преименована је у kocka (слика 6.38.).

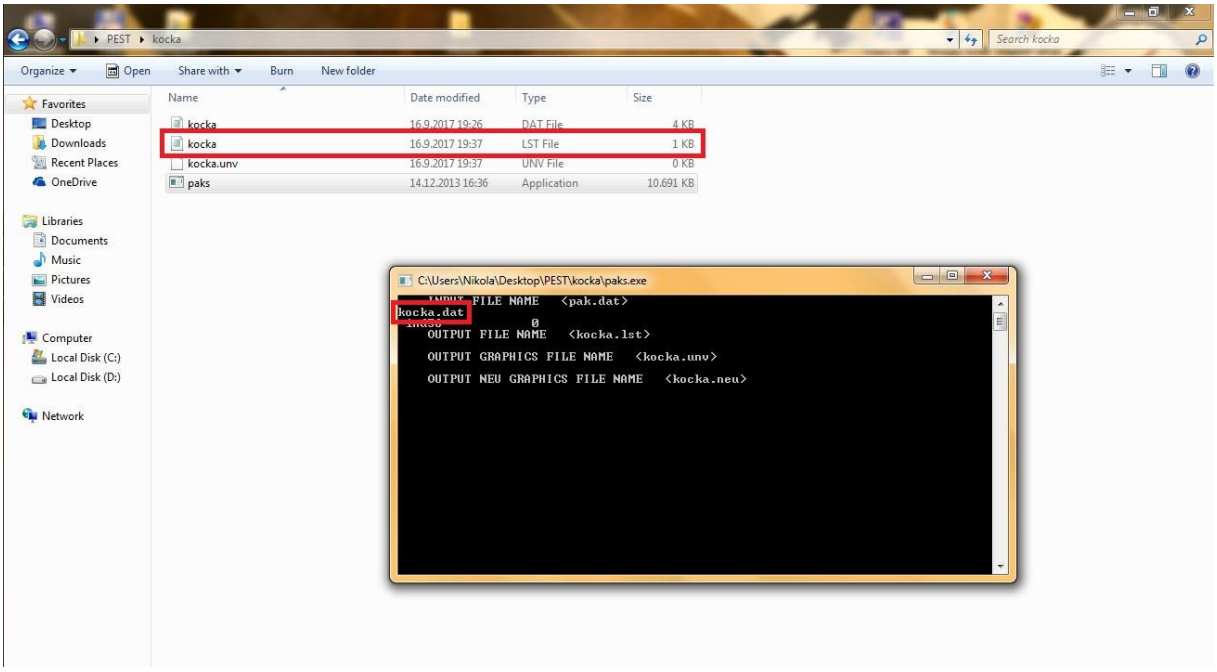



*Слика 6.38. Улазна датотека прорачуна модела у PEST терминалу*

У излазној датотеци прорачуна коју генерише PAK, када се анализа изврши посредством FEMAP-а, не могу се видети померања чворова, као и напони који се јављају у чворовима. До овог проблема долази услед аутоматских подешавања програма. Потребно је извршити малу корекцију у улазној датотеци (као на слици), што ће решити овај проблем, и након покретања прорачуна са оваквом улазном датотеком, у излазној датотеци биће одштампана померања и напони у чворовима (слика 6.39.).



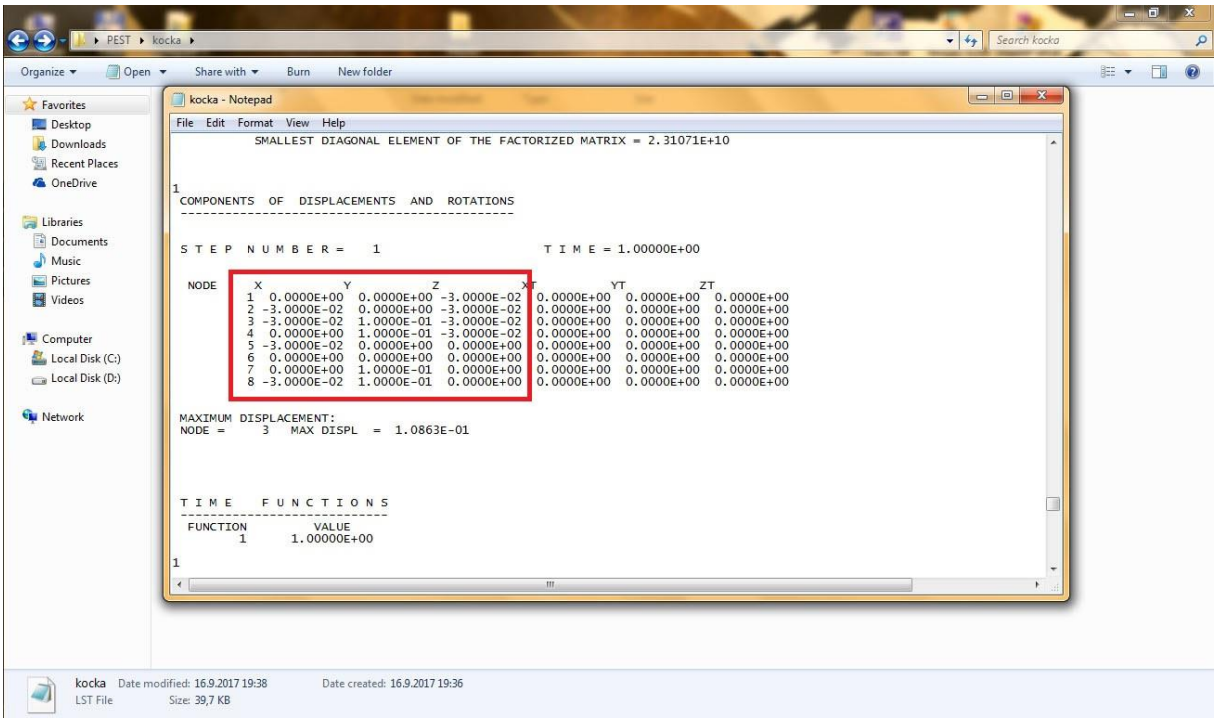

*Слика 6.39. Измена датотеке kocka.dat*




*Слика 6.40. Креирање излазне датотеке kocka.lst*

Када је корекција улазне датотеке извршена, потребно је покренути апликацију paks, и када се прозор отвори унети назив улазне датотеке (у овом случају kocka.dat) (слика 6.40.).

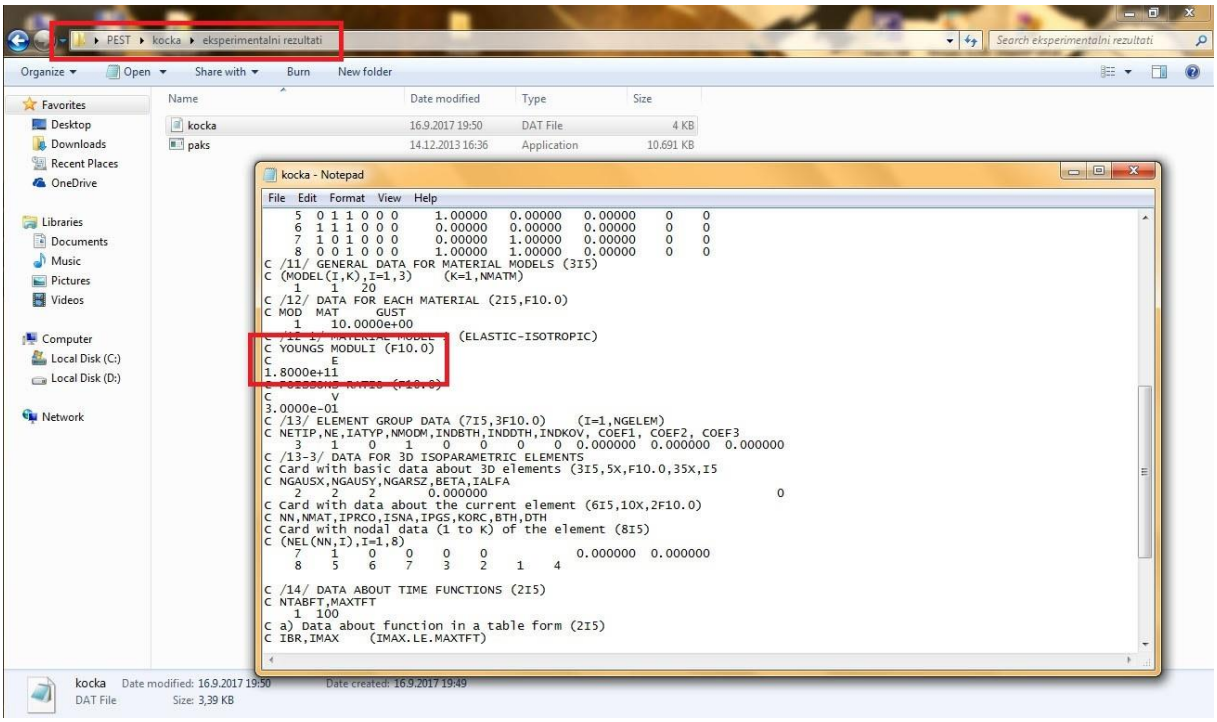
Извршавањем ове команде у истом директоријуму биће генерисана излазна датотека kocka.lst. У овако добијеној излазној датотеци могу се наћи вредносто померања чворова услед дејства сила у чворовима (приказано на слици 6.41.), као и напона у чворовима. У овом примеру биће коришћене вредности померања чворова.




*Слика 6.41. Структура излазне датотеке kocka.lst*

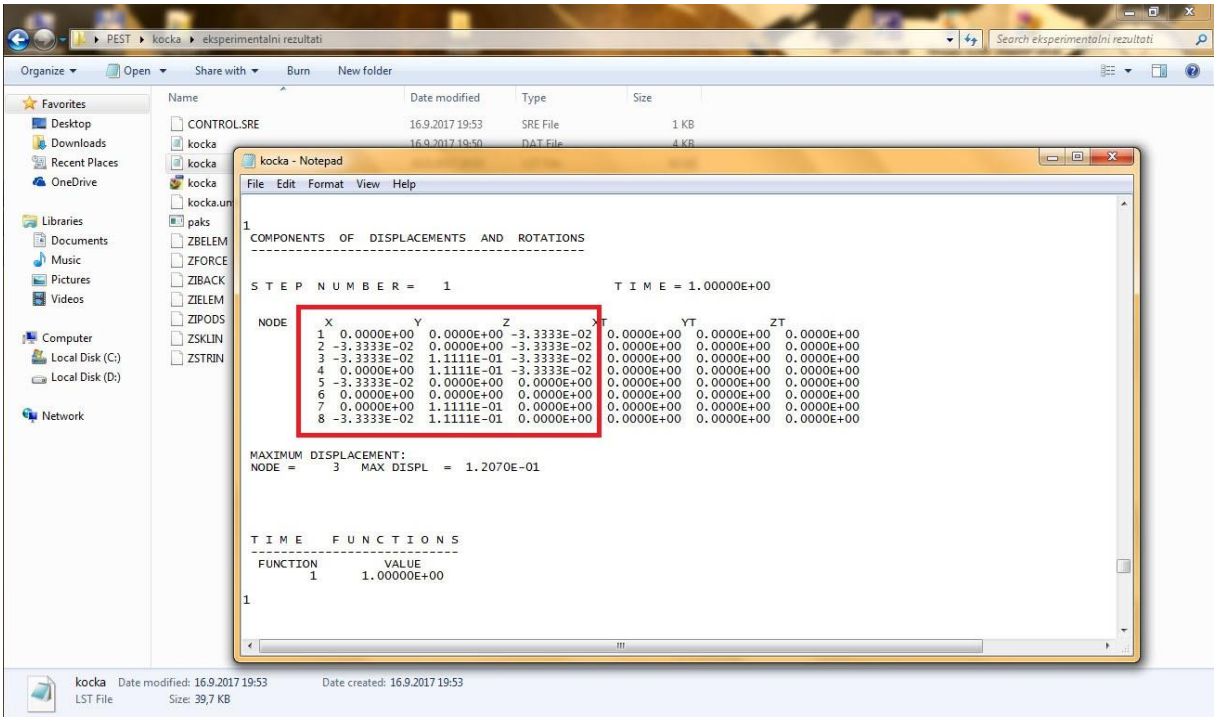
Претходно приказан излазни директоријум добијен је када се прорачун изврши са параметрима какви су приказани на слици, при чему треба обратити пажњу на вредност Јанговог модула еластичности који у датом случају износи $E = 2E11$.

Следеће је потребно креирати нови директоријум (у овом примеру назван експерименталне вредности) у коме ће се прорачун извршити још једном, али сада са промењеним модулом еластичности који ће износити $E = 1.8E11$ (слика 6.42.).




*Слика 6.42. Измена вредности модула еластичности улазне датотеке у терминалу експерименталне вредности*

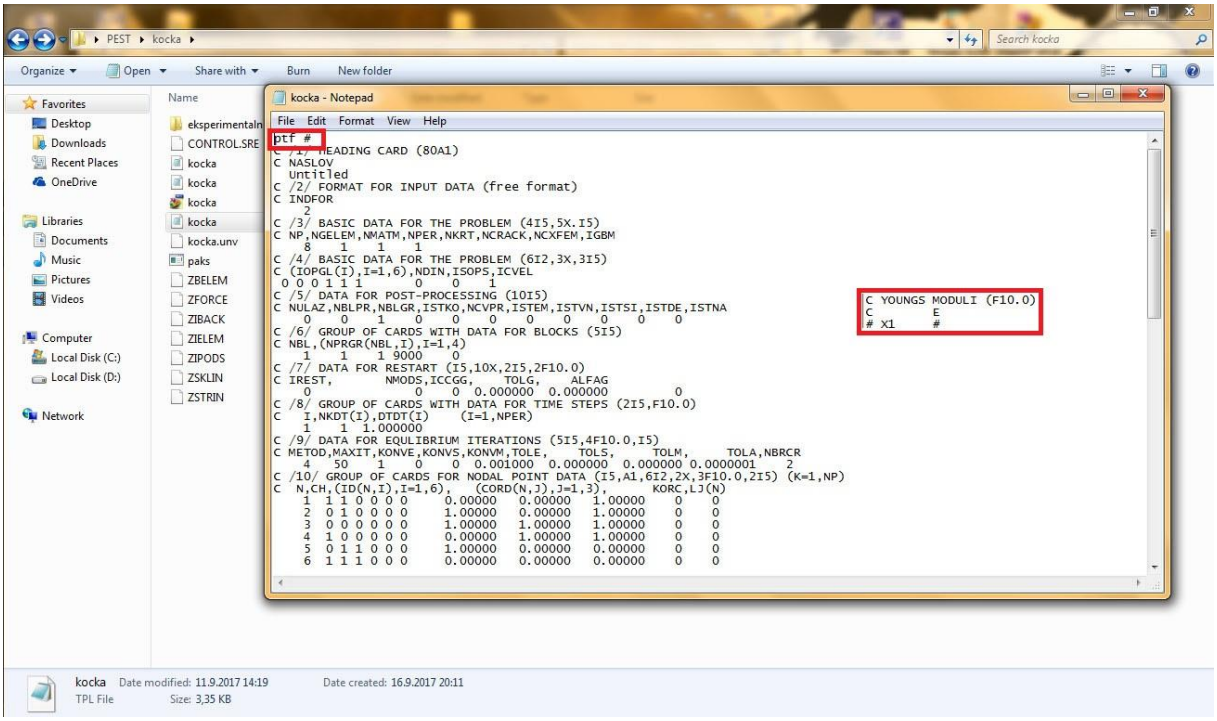
Када се прорачун са оваквим вредностима улазних параметара изврши, добијају се резултати померања чворова као на слици. Ове вредности померања чворова ће служити као циљани излаз при оптимизацији (слика 6.43.).




*Слика 6.43. Вредности померања чворова у излазној датотеци креираној у терминалу експерименталне вредности*

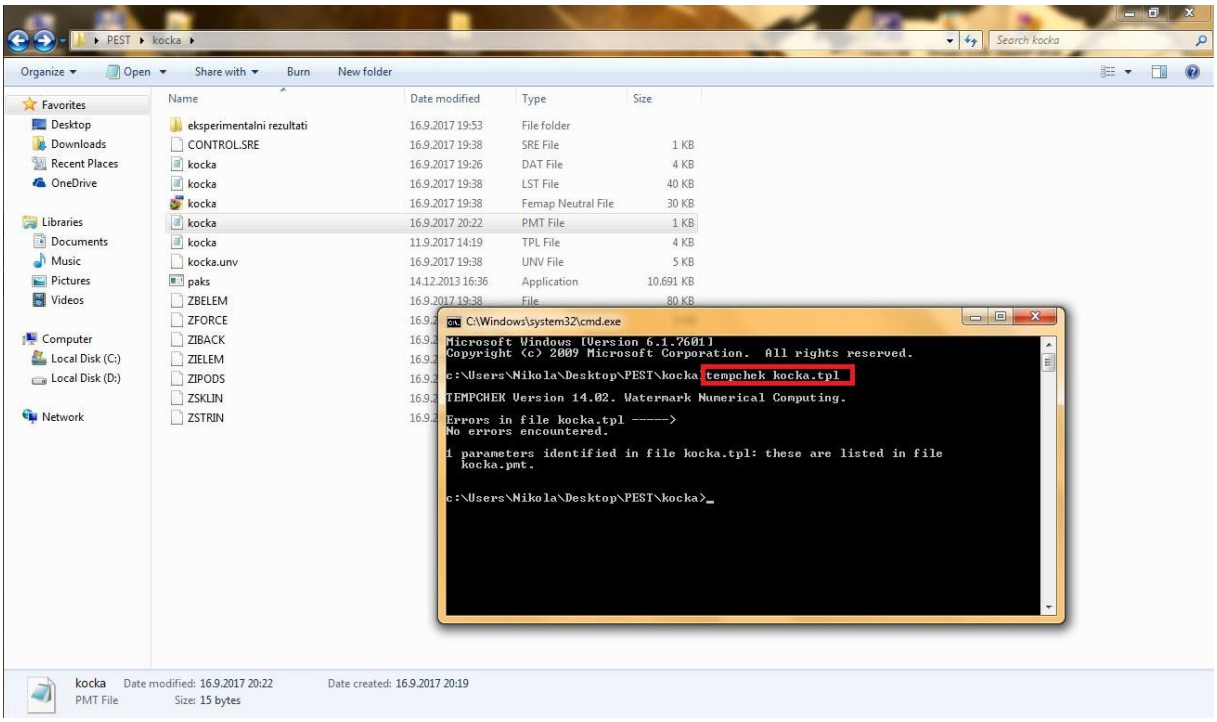
За вредности циљаних излаза у пракси се могу користити и измерени експериментални резултати на реалном моделу, ако се као пример који се подвргава оптимизацији исти тај модел софтверски креира. Након што су креиране све улазне и излазне датотеке, приступа се креирању датотека које ће PEST-у давати инструкције које параметре и на основу чега оптимизовати.

Прво је потребно креирати PEST Template File. Структура Темплате датотеке је детаљно објашњена у претходном примеру. Јангов модул еластичности је замењен ознаком x1 и стављен између маркера (#). Посебну пажњу треба обратити да простор који је резервисан за испис вредности модула еластичности буде обухваћен маркерима (као на слици). Овако креирану датотеку потребно је сачувати под екстензијом .tpl. (слика 6.44.).



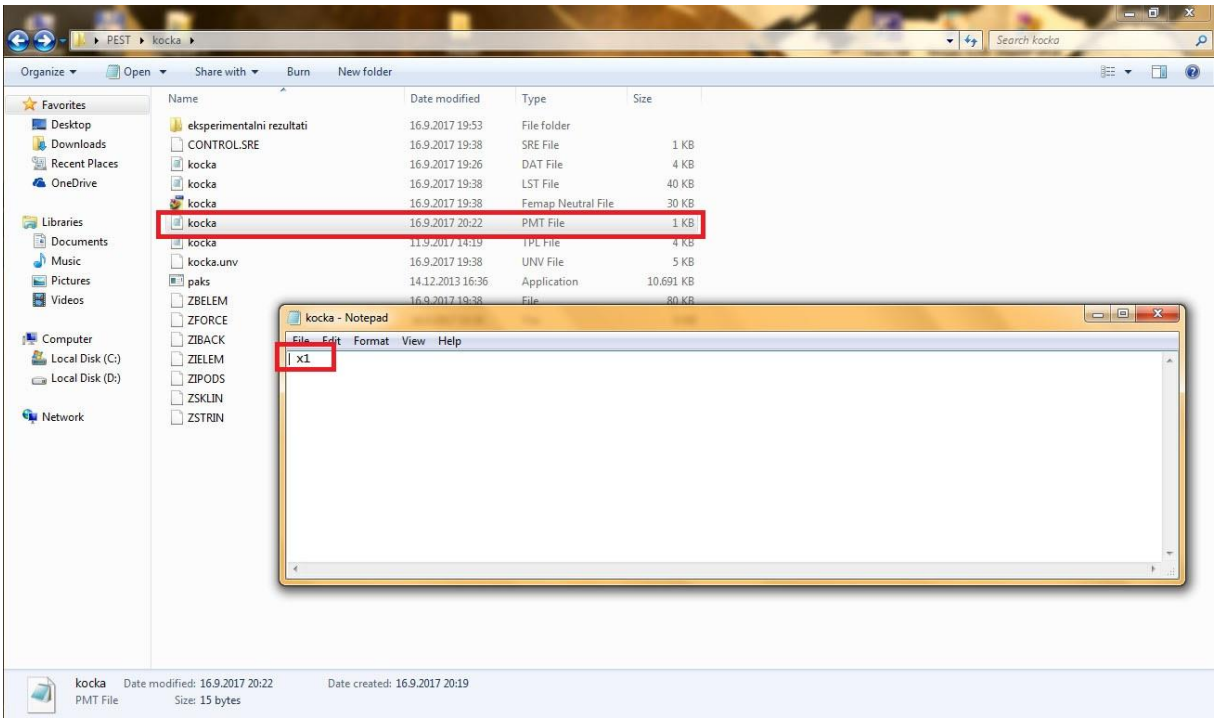

*Слика 6.44. Структура датотеке kocka.tpl*

Следеће што је потребно урадити јесте проверити да ли је Template датотека коректно креирана. То се ради тако што се покрене command prompt у терминалу где је смештена Temlate датотека и у њему се изда команда pestchek kocka.tpl. (слика 6.45.).



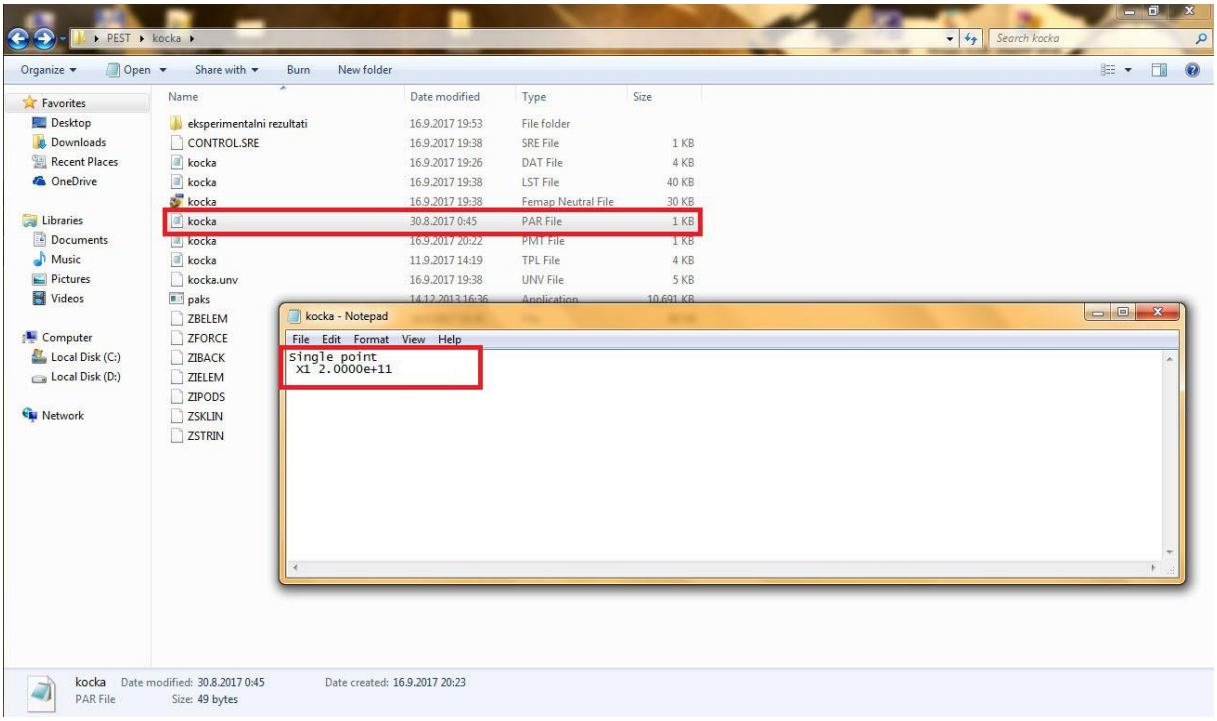

*Слика 6.45. Провера коректности датотеке kocka.tpl*

Уколико је Template директоријум коректно креиран аутоматски ће бити креирана датотека kocka.pmt у којој ће бити излистани параметри који су маркирани у Template датотеци као на слици 6.46. (у овом примеру то је параметар x1 који је додељен Јанговом модулу еластичности).




*Слика 6.46. Структура аутоматски креиране датотеке kocka.pmt*

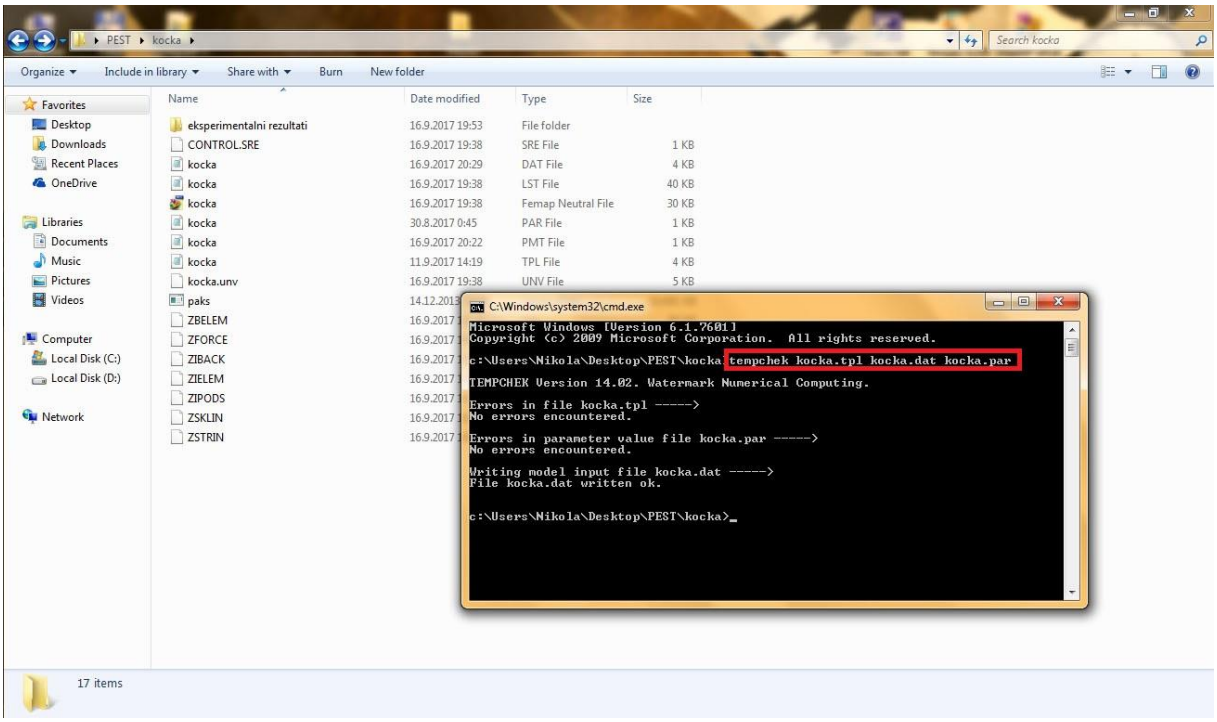
Следећа датотека коју је потребно креирати је kocka.par. У овој датотеци дефинише се вредност од које ће PEST почети варирање датог параметра. У овом случају задата је иста вредност која су коришћена у улазној датотеци за модул еластичности, E = 2E11. (слика 6.47.).



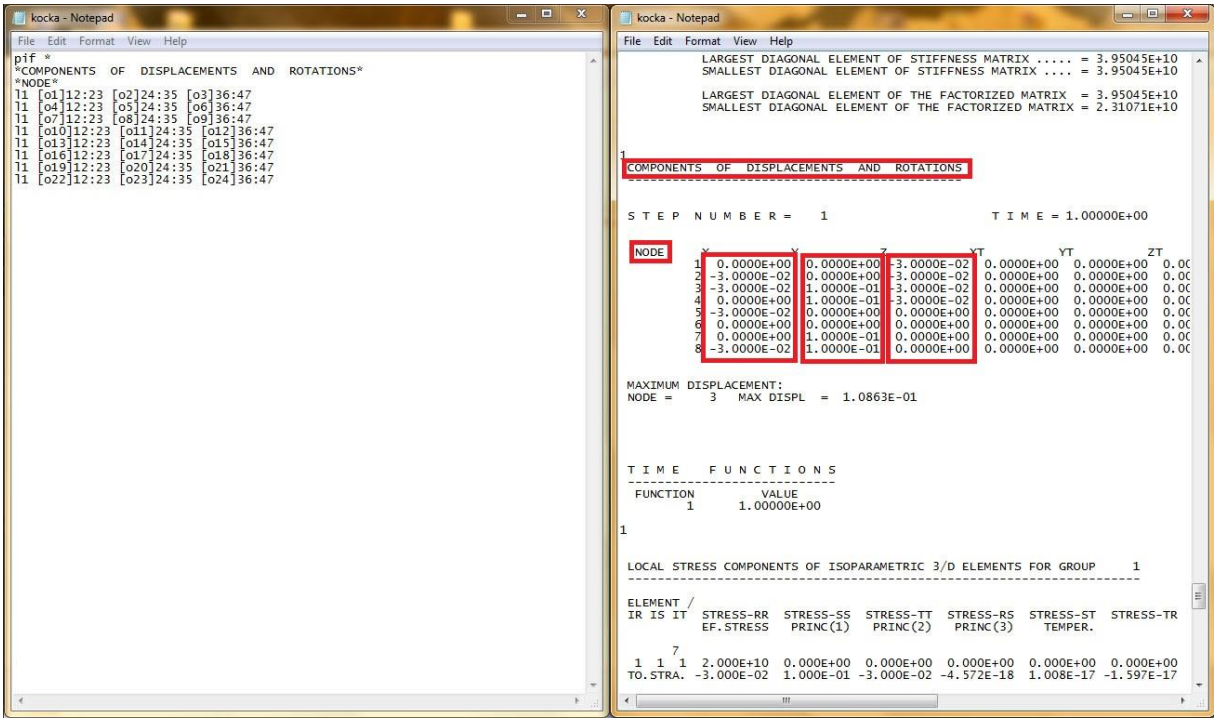

*Слика 6.47. Структура датотеке kocka.par*

Када су све наведене датотеке креиране, потребно је проверити коректност истих. Потребно је покренути command prompt у терминалу где су датотеке смештене и унети команду tempchek kocka.tpl koka.dat koka.par. Након овога, у случају да је дошло до грешке приликом креирања неке од датотека, програм ће јавити грешку, и у којој датотеци се налази грешка. Ако је све коректно креирано може се наставити даље (слика 6.48.).

Следећи сет датотека који је потребно креирати су инструкционе датотеке. Прво је потребно креирати PEST Instruction File. У овој датотеци потребно је дати инструкцију PEST-у како да маркира вредности излаза из излазне датотеке kocka.lst. У првој линији кода датотеке kocka.lst потребно је унети pif као што је објашњено у претходном примеру. Симболима (#) означавају се карактеристични редови у излазној датотеци како би PEST препознао од ког реда креће да ишчитава вредности. Након тога потребно је унети број поља где је смештена вредност, тј. колико поља заузима вредност која треба да буде учитана као што је приказано на слици 6.49.



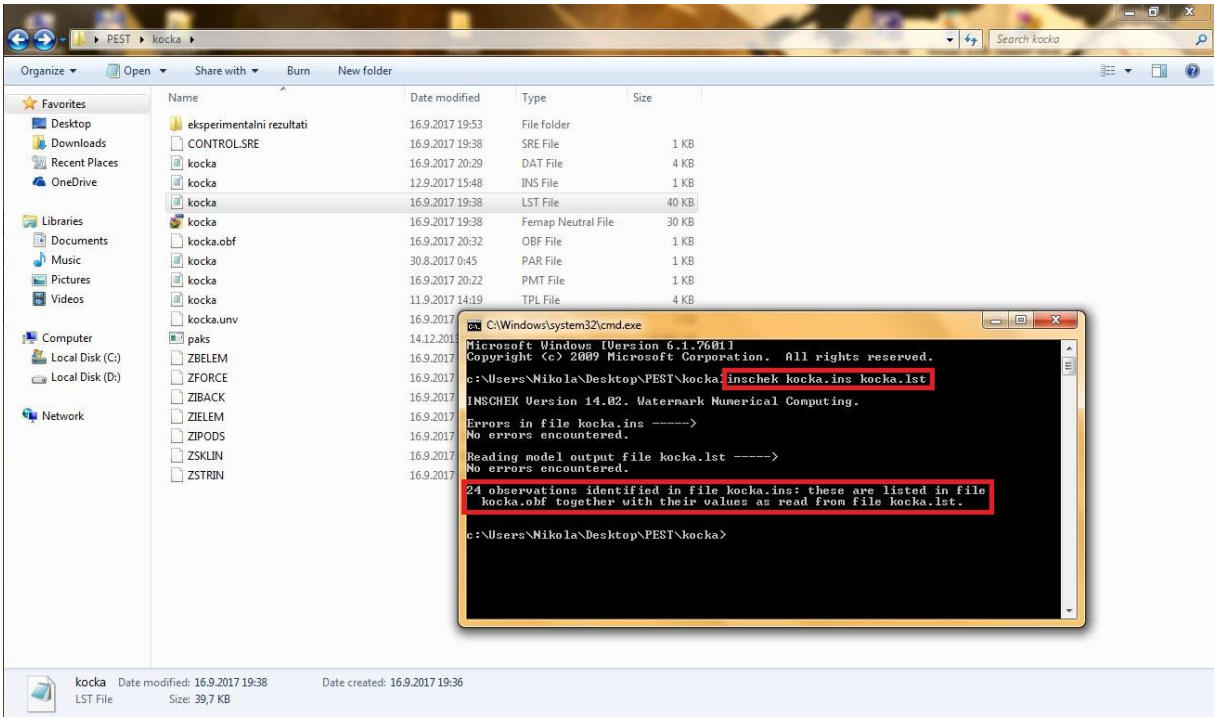

*Слика 6.48. Провера коректности претходно креираних датотека*




*Слика 6.49. Структура датотеке kocka.tpl*

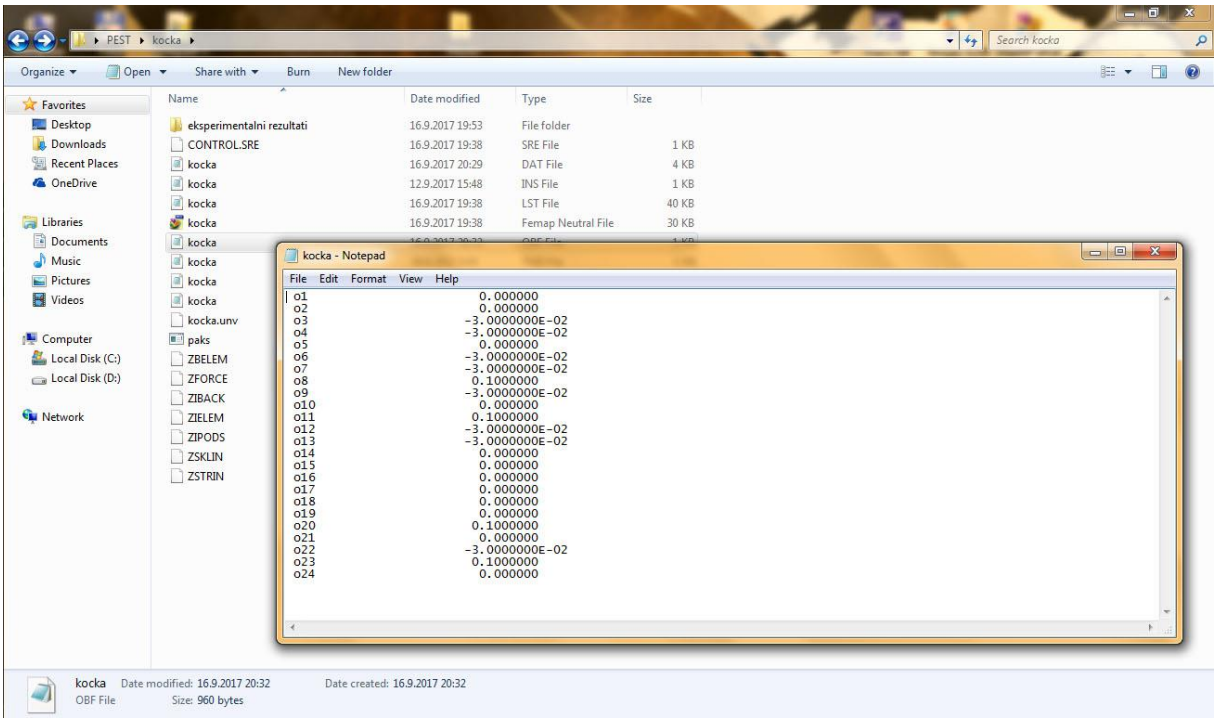
Након креирања ове датотеке, потребно је покренути command prompt, и издати команду inschek kocka.ins. На овај начин, аутоматски ће се креирати датотека kocka.obf у којој ће редом бити излистани симболи којима се означавају редни бројеви вредности које требају да буду учитане из излазне датотеке. Затим, у command prompt-у уносимо

инструкцију inschek kocka.obf kocka.lst, којом ће у датотеци kocka.obf бити излистане вредности померања чворова из излазне датотеке kocka.lst. (слика 6.50.).



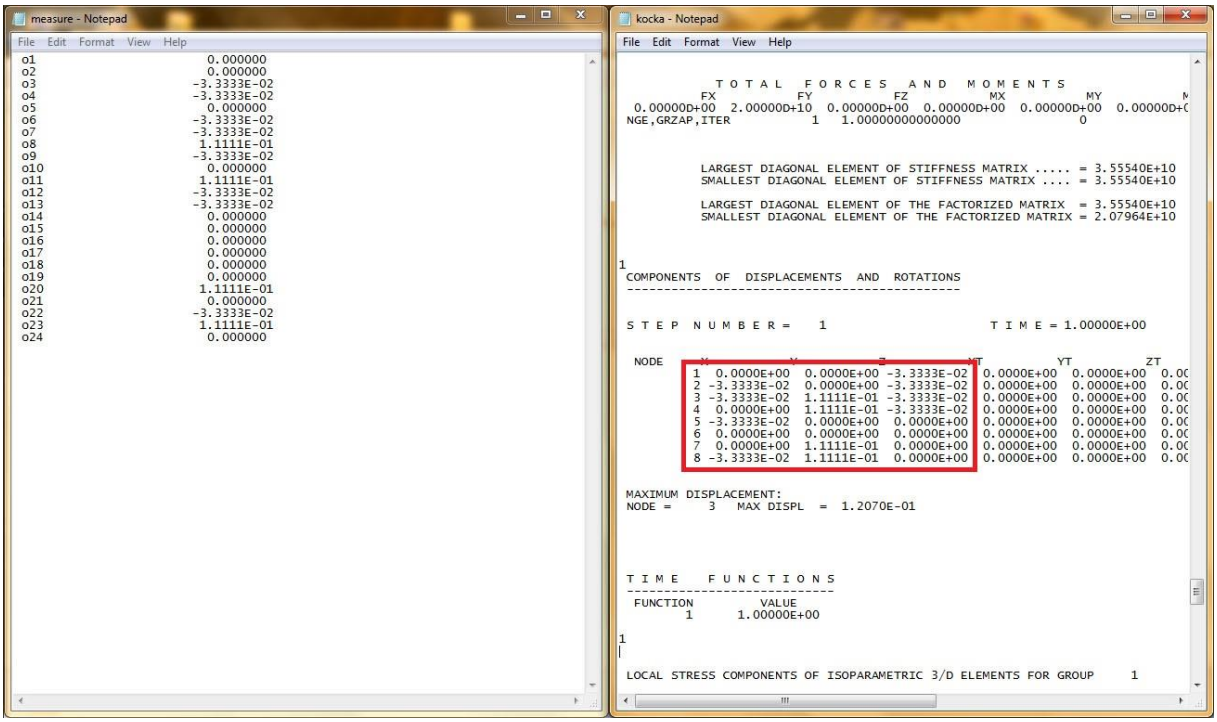

*Слика 6.50. Генерисање датотеке kocka.obf*

На слици су приказане вредности померања чворова модела које је PEST ишчитао у датотеци kocka.obf. (слика 6.51.).

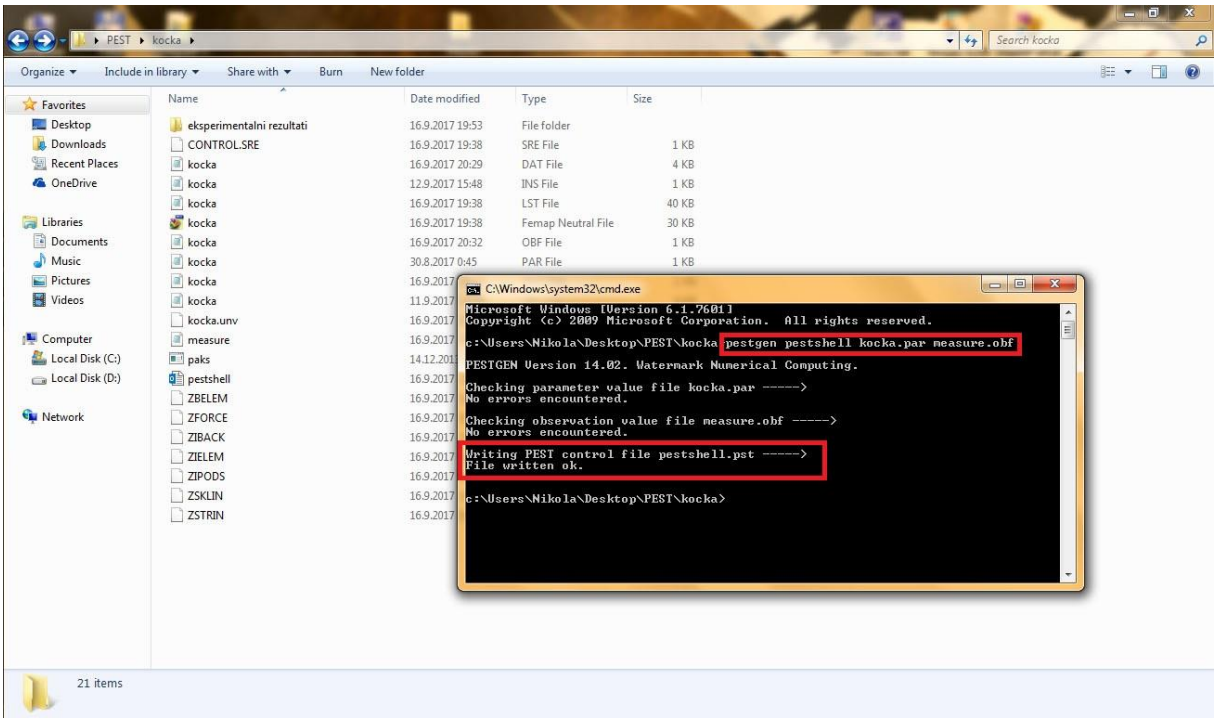



*Слика 6.51. Структура датотеке kocka.obf*

Следећи корак је креирање measure.obf датотеке, која служи да PEST-у обезбеди вредности са којима ће поредити резултате прорачуна при свакој новој итерацији. У овом случају за вредности које су унешене у measure.obf, узимају се вредности излазних параметара из прорачуна који је извршен у директоријуму експериментални резултати као што је приказано на слици 6.52.




*Слика 6.52. Структура датотеке measure.obf*




*Слика 6.53. Генерисање датотеке pestshell.pst*

Након што је креирана датотека measure.obf, потребно је генерисати финалну датотеку која ће објединити све инструкције и параметре из свих претходно креираних датотека. То се ради покретањем command prompt-а у терминалу где су смештене све претходно креиране датотеке, и издавањем команде pestgen pestshell kocka.par measure.obf. pestshell је име датотеке коју ће PEST генерисати у овом случају (слика 6.53.). Овако креирана датотека има екстензију .pst и помоћу ње ће се вршити оптимизација.

Да би pestshell могао да извршава своју функцију мора постојати извршни exe који покреће прорачун самим његовим покретањем. Paks није у могућности да изврши прорачун док му се не унесе име улазне датотеке коју користи. Да би се решио овај проблем потребно је написати програм или скрипту који ће позивати paks затим исписати име улазне датотеке и које излазне датотеке треба да креира.

Један од најлакших начина да се ово уради је да се напише скрипта у notepad-у која ће извршавати претходно речено. У notepad-у је потребно исписати

paks < komande > provera

а затим то снимити под екстензијом .bat (у конкретном примеру runpak.bat). Ова скрипта покреће paks, узима команде из спољне датотеке komande, и исписује оно што је програм извршио у датотеку provera [13].

У датотеци команде треба унети следеће:

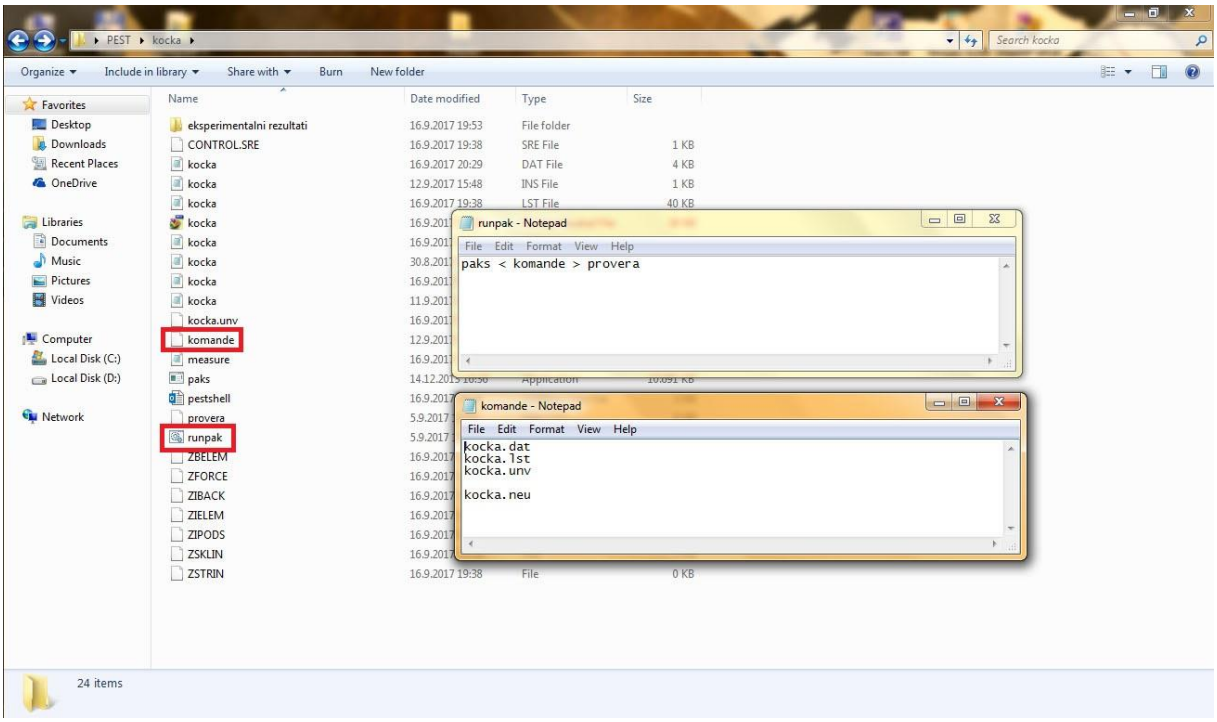
kocka.dat

kocka.lst

kocka.unv

kocka.neu

и снимити без икакве екстензије. kocka.dat је улазна датотека, а остале три су излазне датотеке које paks исписује (слика 6.54.).[13].




*Слика 6.54. Структура скрипте runpak*

Када је скрипта конфигурисана, потребно је унети измене у датотеци pestshell. Као на слици, уместо постојећег потребно је унети:

* model command line

runpak

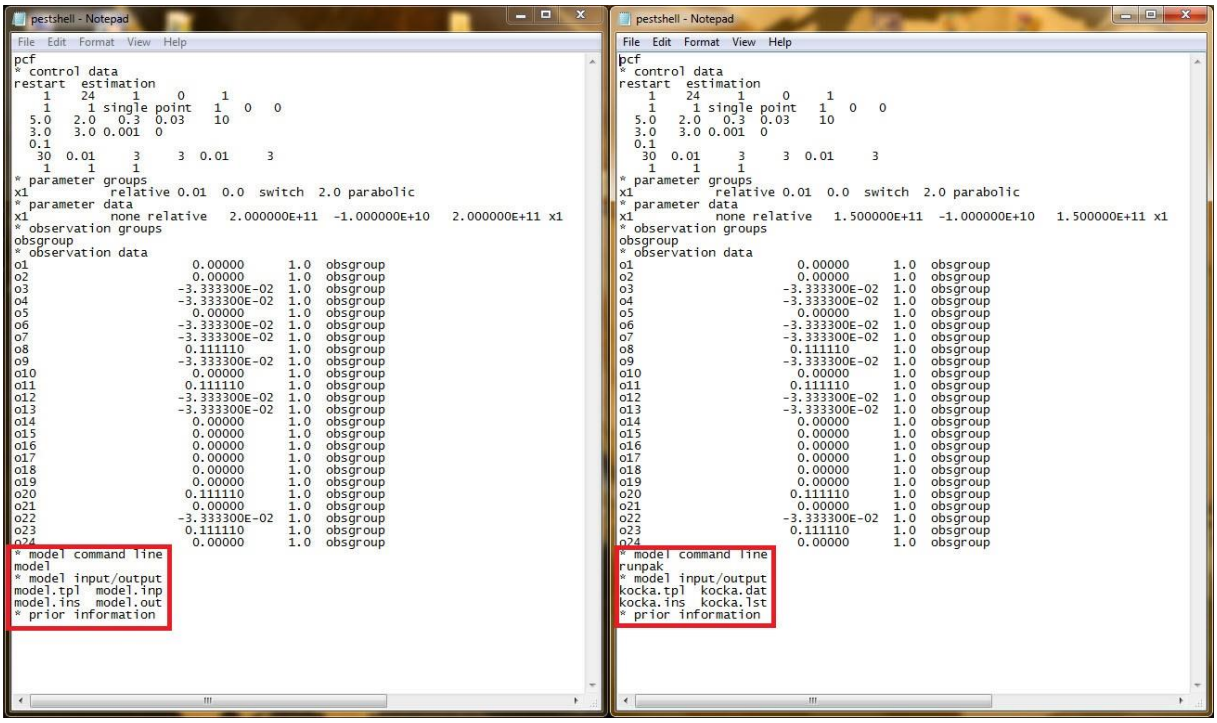
* model input/output

kocka.tpl  kocka.dat

kocka.ins  kocka.lst

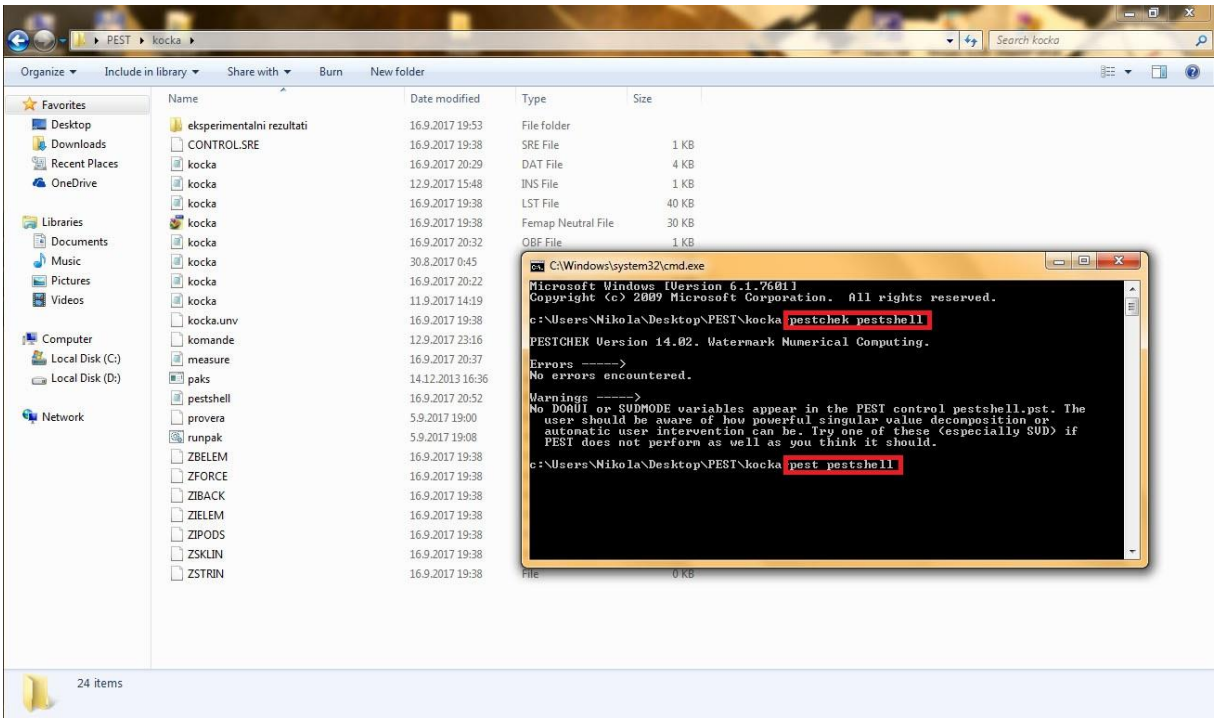
* prior information

Аналогно претходном примеру, за извршни програм треба поставити runpak, као улазну и излазну датотеку kocka.dat и kocka.lst, и као template и instruction датотеке kocka.tpl и kocka.ins. Овим је завршена измена pestshell датотеке (слика 6.55.).



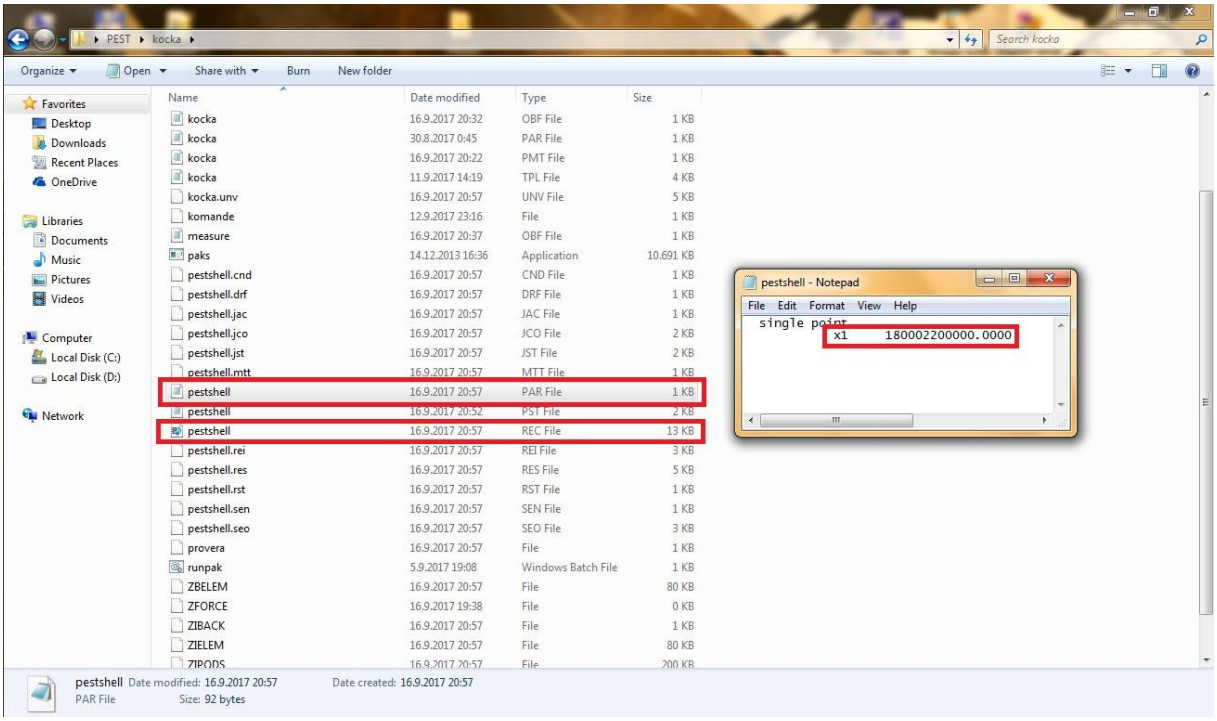

*Слика 6.55. Измена датотеке pestshell.pst*

Сада је све спремно за покретање саме оптимизације. Прво треба проверити да ли је директоријум pestshell коректно креиран командом pestchek pestshell. Ако јесте је покренути command prompt у директоријуму kocka и унети наредбу pest pestshell. На овај начин покреће се оптимизација (слика 6.56.).




*Слика 6.56. Покретање PEST оптимизације*

Након извршеног прорачуна, PEST генерише pestshell.par датотеку у којој су смештени резултати оптимизације. На слици 6.57. се може видети да у овом случају резултат оптимизације изузетно мало одступа од референтне вредности модула еластичности (мање од 1%) па се резултати оптимизације могу сматрати јако повољним.




*Слика 6.57. Резултат PEST прорачуна у датотеци pestshell.par*

У датотеци pestshell.rec исписан је цео поступак прорачуна оптимизације. Овде се може наћи колико је итерација извршено, које вредности су додељиване траженим параметрима приликом сваке итерације.

У конкретном случају, у датотеци kocka.rec, може се видети да је извршено 7 итерација, тј.7 пута је прорачун био извршен са различитим вредностима које је PEST додељивао моделу за модул еластичности све док није дошао до вредности за коју прорачун даје задовољавајуће резултате.

## 6.5. Идентификација параметара материјалних карактеристика модела када је материјал нелинеаран

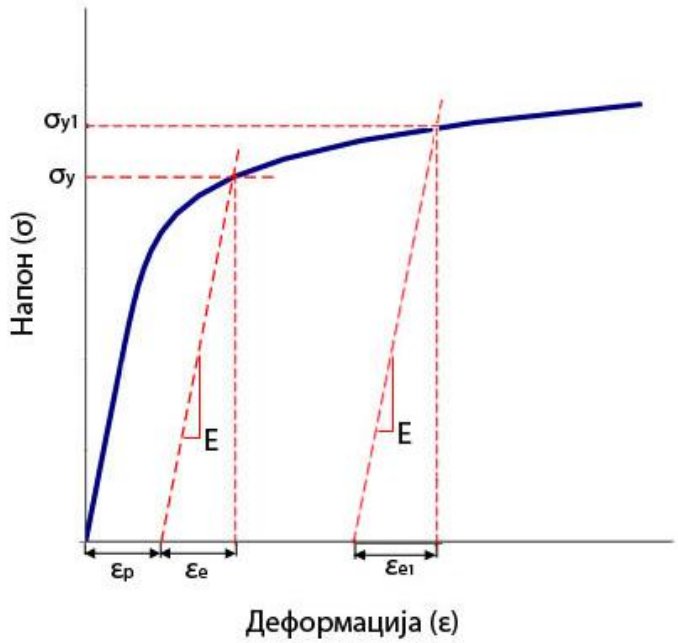
У претходном примеру рађена је идентификација материјалних параметара линеарног модела. Оптимизациони пакет PEST је у том случају веома брзо пронашао циљане параметре. Следећи пример је рађен са истим моделом, с тим што су материјалу задати параметри за нелинеарност.

Поред Јанговог модула еластичности E=2e9 и Поасонвог коефицијента V=0,3, материјалу су задати и напон течења TAUY=2e8 и модул пластичности Cy=2e8. Параметар AN утиче на линеарност деформације када материјал уђе у зону пластичних деформација.

У случају овако дефинисаног материјала, функција деформација у зависности од напона описана је *Ramberg-Osgood*-овом једначином.

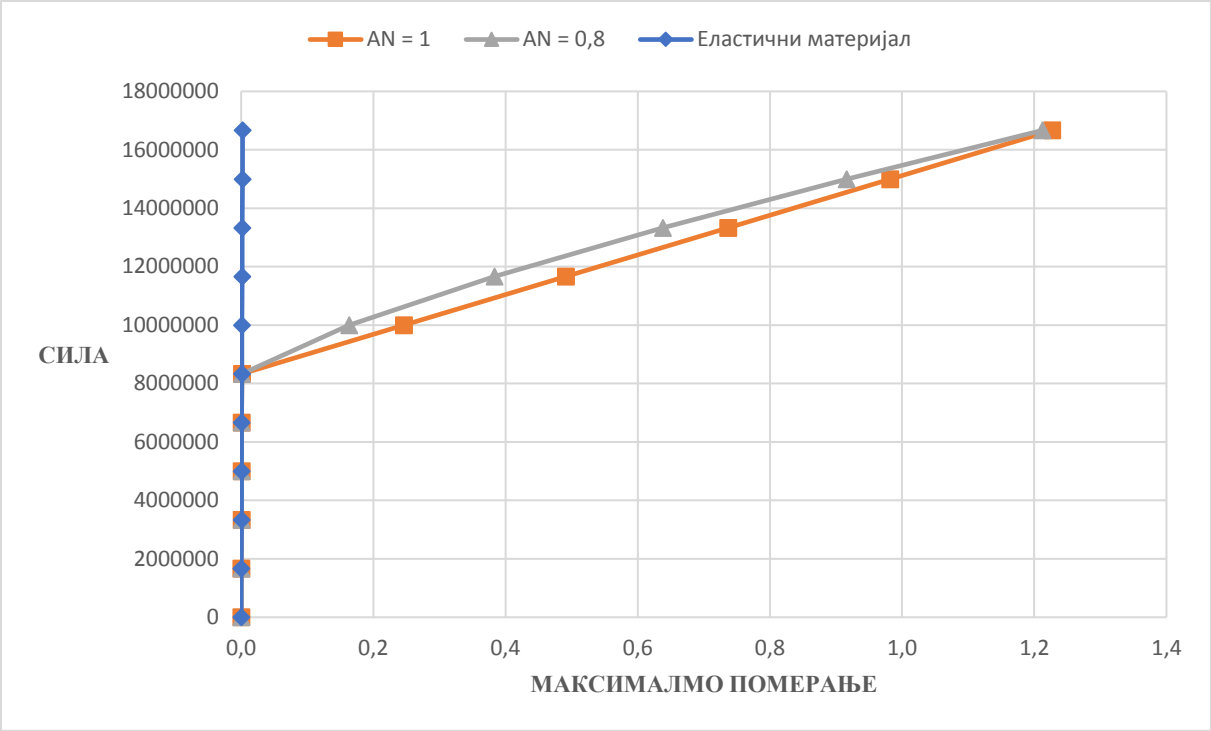
$$\sigma = \sigma_y + C_y \varepsilon_p^n$$

На слици 6.58. приказана је зависност деформација од напона. Може се приметити да се са порастом напона јавља пораст и еластичних и пластичних деформација.



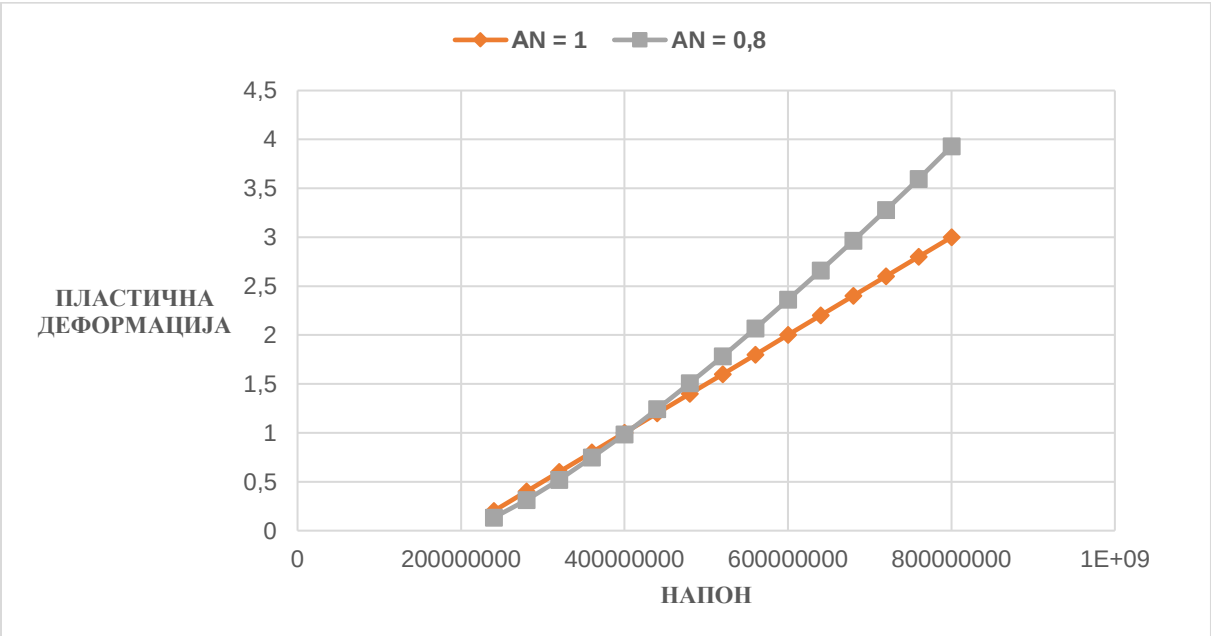

*Слика 6.58. Зависност еластичних дегормација од напона [15]*

Тестирана су два случаја, када је AN=1 материјал се у зони пластичности деформише линеарно, и када је AN=0,8 где долази до нелинеарних деформација. На слици 6.58. је приказан дијаграм на коме се види разлика у померању чворова под дејством сила код линеарног, еластичног материјала, материјала када је AN=1 и када је AN=0,8.




*Слика 6.58. Померања чворова модела у зависности од силе којом је модел оптерећен*

На слици 6.59 може се приметити разлика деформација у зони пластичности материјала у зависности од параметра AN.




*Слика 6.59. Зависност пластичних деформација од напона у материјалу*

### 6.5.1. Идентификација параметара материјалних карактеристика нелинеарног модела AN=1

У овом примеру биће извршена идентификација параметара који описују нелинеарност материјала. Параметар AN=1 имплцира да су деформације материјала у зони пластичности линеарне. Анализа се врши у 30 интервала, и за сваки се добија померање чворова и напони при силама које у том интервалу делују на модел.

Код овог модела тражени су следећи параметри: модул еластичности E=2e11, Поасонов коефицијент V=0,3, напон течења TAUY=2e8, модул пластичности Cy=2e8 и AN=1. Као вредности од којих PEST креће са прорачуном узете су вредности: TAUY=1,5e8, Cy=1,5e8, AN=0,9, V=0,3 и E=1,5e11.

За вредности са којима PEST током прорачуна пореди резултате узете су вредности померања чворова по у оси, и то за 1., 10., 20. и 30. интервал, а формиране су прорачуном извршеним са горе поменутим вредностима са којима PEST креће са идентификацијом.

PEST је за идентификацију параметара модела извршио 3 итерације и покренуо прорачун у PAK-у 41 пут. То значи да је 41 пут мењао тражене параметре у улазној датотеци модела и са тако формираним улазом покретао прорачун.

Вредности материјалних параметара модела које је PEST идентификовао су следеће: TAUY=199977063, Cy=200009900 и AN=0,99990487. Модул еластичности и Поасонов коефицијент PEST није успео да пронађе.

### 6.5.2. Идентификација параметара материјалних карактеристика нелинеарног модела AN=0,8

Као и у претходном, и у овом примеру биће извршена идентификација параметара који описују нелинеарност материјала. Параметар AN=0,8 имплцира да су деформације материјала у зони пластичности нелинеарне. Анализа се врши у 30 интервала, и за сваки се добија померање чворова и напони при силама које у том интервалу делују на модел.

Код овог модела тражени су следећи параметри: модул еластичности E=2e11, Поасонов коефицијент V=0,3, напон течења TAUY=2e8, модул пластичности Cy=2e8 и AN=0,8. Као вредности од којих PEST креће са прорачуном узете су вредности: TAUY=1,5e8, Cy=1,5e8, AN=0,9 и V=0,3 и E=1,5e11.

За вредности са којима PEST током прорачуна пореди резултате узете су вредности померања чворова по у оси, и то за 1., 10., 20. и 30. интервал, а формиране су прорачуном извршеним са горе поменутим вредностима са којима PEST креће са идентификацијом.

PEST је за идентификацију параметара модела извршио 11 итерација и покренуо прорачун у PAK-у 65 пута. То значи да је 65 пута мењао тражене параметре у улазној датотеци модела и са тако формираним улазом покретао прорачун.

Вредности материјалних параметара модела које је PEST идентификовао су следеће: TAUY=20178277, Cy=198220287 и AN=0,804414086. Вредности параметара E и V, PEST није успео да пронађе.

### 6.5.3. Анализа резултата

Обзиром да оптимизационим процесом нису коректно идентификовани сви тражени параметри, може се закључити да одређени параметри процеса нису правилно подешени. Параметри карактеристични за пластичност материјала су веома прецизно пронађени насупрот параметрима карактеристичним за еластичност.

Прво што се може уочити је да су поредбене вредности померања чворова на 1., 10., 20. и 30. кораку анализе. Сем вредности померања на 1. кораку све остале вредности померања налазе се дубоко у зони пластичности. Ово се може решити са више поредбених вредности у зони еластичности што у претходним примерима чине првих 5 корака анализе.

Такође, из претходно виђене криве напон-деформација добијене *Ramberg-Osgood*-овом једначином може се закључити да је у поредбене параметре потребно укључити и пластичне деформације у истим корацима у којима се посматрају и укупне деформације. Виђено је да са порастом напона долази и до повећања еластичних деформација, тако да се из тога може закључити да поред укупних померања морају да се прате и пластичне деформације како би се квалитетно идентификовали параметри карактеристични за еластичност.

Овимм се може закључити да је за прецизно идентификовање параметара материјала потребно добро познавати карактеристике материјала као и међусобне зависности свих његових параметара.

# 7. Закључак

Оптимизација налази свакодневну примену у покушајима да се са што мање ангажовања постојећих ресурса постигне што је могуће већи ефекат у процесима планирања производње, алоцирања ресурса, временског планирања, управљања системима, одлучивања и слично. Не постоји општи поступак за решавање произвољног оптимизационог проблема, па је временом развијен већи број метода у складу са природом третираног оптимизационог проблема. За добијање најбољих резултата оптимизације јако је битно којом од метода се оптимизација врши.

Метод коначних елемената (МКЕ) је данас најопштији нумерички метод, примењен у готово свим наукама, а посебно у инжењерским областима. Основне карактеристике МКЕ које га доводе испред других метода се састоје у његовој општости (применљив је на област солида, флуида, провођења топлоте, и уопште на проблеме поља физичких величина, као и на спрегнуте проблеме), применљивости на линеарне и нелинеарне проблеме, релативној једноставности у основним поставкама, као и погодности у погледу примене рачунара.

У раду је презентована идентификација материјалних параметара FEM модела вршена програмским пакетом PEST. Детаљно су објашњене PEST процедуре које треба извршити и специфичности проблема овакве оптимизације. Примером идентификације параметара модела који је овим радом презентован, показано је да је помоћу PEST-а уз одређене процедуре прилагођавања појединих извршних датотека, могуће вршити идентификацију параметара у анализи (МКЕ) вршеној у програму PAK. Идентификација одређених параметара FEM модела је веома применљива у индустрији за решавање реалних проблема и уштеду времена и ресурса.

# Литература

[1] Singiresu S. Rao, Engineering Optimization: Theory and Practice, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[2] Никола Миливојевић, Оптимизационе методе у симулацији у управљању хидроенергетским системима, Докторска теза, Машински факултет, Универзитета у Крагујевцу, 2008.

[3] Тина Дашић, Милош Станић, Методе оптимизације: Методе глобалне оптимизације, Грађевински факултет Универзитета у Београду, 2014., (Презентације са предавања)

[4] Александар Д. Купусинац, Поређење алгоритама комбинаторне оптимизације, 18. Телекомуникациони форум TELFOR 2010, Србија, Београд, новембар 23.-25., 2010.

[5] http://www.sspa.com/software/pest, (приступљено 11. 10. 2017.)

[6] http://www.pesthomepage.org/, (приступљено 11. 10. 2017.)

[7] Model-Independent Parameter Estimation User Manual Part I: PEST, SENSAN and Global Optimisers, 2016., (Корисничко упутство)

[8] http://itcr.hr/femap/femap/, (приступљено 11. 10. 2017.)

[9] https://www.plm.automation.siemens.com/en/products/femap/features/ (приступљено 11. 10. 2017.)

[10] Милош Којић, Радован Славковић, Мирослав Живковић, Ненед Грујовић: Метод коначних елемената 1, Машински факултет, Крагујевац, 1998.

[11] https://www.java.com/en/download/help/path.xml, (приступљено 11. 10. 2017.)

[12] Building Your First Models in FEMAP, Network Analysis, Inc., W. Lindbergh Way, Chandler, Arizona, January 2004., (Корисничко упутство)

[13] https://www.tutorialspoint.com/batch_script/index.htm, (приступљено 11. 10. 2017.)

[14] https://en.wikipedia.org/wiki/Work_hardening, (приступљено 20. 10. 2017.)

[15] https://en.wikipedia.org/wiki/Ramberg%E2%80%93Osgood_relationship, (приступљено 20. 10. 2017.)

# Прилози

**Прилог 1.** Код пограмског решења eksponencijalna funkcija.

```
      program eksponencijalna funkcija
      integer*4 i,nx
      real*4 y1,xc
      real*4 x(50),y(50)
      write(6,10)
10    format(/,' Program EKSPONENCIJALNA FUNKCIJA. Watermark Computing')
      write(6,20)
20    format(/,' Reading model input file IN.DAT....')
      open(unit=20,file='in.dat')
c     read the line parameters
      read(20,*) y1
      read(20,*) xc
c     read the abscissae at which there are measurement values
      read(20,*) nx
      do 100 i=1,nx
      read(20,*) x(i)
100   continue
      close(unit=20)
      write(6,110)
110   format(' File IN.DAT read ok.')
c     evaluate y for each x
      do 200 i=1,nx
        y(i)=x(i)**xc+y1
      end
200   continue
c     write the y values to the output file
      write(6,220)
220   format(/,' Writing model output file OUT.DAT...')
      open(unit=20,file='out.dat')
      do 300 i=1,nx
      write(20,*) x(i),y(i)
300   continue
      close(unit=20)
      write(6,320)
320   format(' File OUT.DAT written ok.')
      end
```

**Прилог 2.** Улазна датотека програмског решења eksponencijalna funkcija, in.dat.

```
0.7
1.8
13
0.052
0.068
0.103
0.128
0.172
0.195
0.230
0.275
0.315
0.332
0.350
0.423
0.488
```

**Прилог 3.** Излазна датотека програмског решења eksponencijalna funkcija, out.dat.

```
 5.20000011E-02  0.70488435
 6.80000037E-02  0.70791620
 0.10300000      0.71671504
 0.12800001      0.72471601
 0.17200001      0.74206793
 0.19499999      0.75273055
 0.23000000      0.77097577
 0.27500001      0.79790366
 0.31500000      0.82501411
 0.33199999      0.83741951
 0.34999999      0.85112017
 0.42300001      0.91252631
 0.48800001      0.97488797
```

**Прилог 4.** Улазна датотека пограмског решења paks, kocka.dat.

```
C /1/ HEADING CARD (80A1)
C NASLOV
 Untitled
C /2/ FORMAT FOR INPUT DATA (free format)
C INDFOR
   2
C /3/ BASIC DATA FOR THE PROBLEM (4I5,5X.I5)
C NP,NGELEM,NMATM,NPER,NKRT,NCRACK,NCXFEM,IGBM
   8    1    1    1
C /4/ BASIC DATA FOR THE PROBLEM (6I2,3X,3I5)
C (IOPGL(I),I=1,6),NDIN,ISOPS,ICVEL
 0 0 0 1 1 1      0    0    1
C /5/ DATA FOR POST-PROCESSING (10I5)
C NULAZ,NBLPR,NBLGR,ISTKO,NCVPR,ISTEM,ISTVN,ISTSI,ISTDE,ISTNA
   0    0    1    0    0    0    0    0    0    0    0
C /6/ GROUP OF CARDS WITH DATA FOR BLOCKS (5I5)
C NBL,(NPRGR(NBL,I),I=1,4)
   1    1    1 9000    0
C /7/ DATA FOR RESTART (I5,10X,2I5,2F10.0)
C IREST,      NMODS,ICCGG,    TOLG,    ALFAG
   0            0    0  0.000000  0.000000        0
C /8/ GROUP OF CARDS WITH DATA FOR TIME STEPS (2I5,F10.0)
C   I,NKDT(I),DTDT(I)    (I=1,NPER)
   1    1  1.000000
C /9/ DATA FOR EQULIBRIUM ITERATIONS (5I5,4F10.0,I5)
C METOD,MAXIT,KONVE,KONVS,KONVM,TOLE,    TOLS,     TOLM,
TOLA,NBRCR
   4   50    1    0    0  0.001000  0.000000  0.000000 0.0000001    2
C /10/ GROUP OF CARDS FOR NODAL POINT DATA (I5,A1,6I2,2X,3F10.0,2I5)
(K=1,NP)
C  N,CH,(ID(N,I),I=1,6),   (CORD(N,J),J=1,3),      KORC,LJ(N)
   1  1 1 0 0 0 0     0.00000   0.00000   1.00000    0    0
   2  0 1 0 0 0 0     1.00000   0.00000   1.00000    0    0
   3  0 0 0 0 0 0     1.00000   1.00000   1.00000    0    0
   4  1 0 0 0 0 0     0.00000   1.00000   1.00000    0    0
   5  0 1 1 0 0 0     1.00000   0.00000   0.00000    0    0
   6  1 1 1 0 0 0     0.00000   0.00000   0.00000    0    0
   7  1 0 1 0 0 0     0.00000   1.00000   0.00000    0    0
   8  0 0 1 0 0 0     1.00000   1.00000   0.00000    0    0
C /11/ GENERAL DATA FOR MATERIAL MODELS (3I5)
C (MODEL(I,K),I=1,3)    (K=1,NMATM)
   1    1   20
```

```
C /12/ DATA FOR EACH MATERIAL (2I5,F10.0)
C MOD  MAT     GUST
   1    10.0000e+00
C /12-1/ MATERIAL MODEL 1 (ELASTIC-ISOTROPIC)
C YOUNGS MODULI (F10.0)
C        E
2.0000e+11
C POISSONS RATIO (F10.0)
C        V
3.0000e-01
C /13/ ELEMENT GROUP DATA (7I5,3F10.0)    (I=1,NGELEM)
C NETIP,NE,IATYP,NMODM,INDBTH,INDDTH,INDKOV, COEF1, COEF2, COEF3
   3    1    0    1    0    0    0    0  0.000000  0.000000  0.000000
C /13-3/ DATA FOR 3D ISOPARAMETRIC ELEMENTS
C Card with basic data about 3D elements (3I5,5X,F10.0,35X,I5
C NGAUSX,NGAUSY,NGARSZ,BETA,IALFA
   2    2    2       0.000000                                  0
C Card with data about the current element (6I5,10X,2F10.0)
C NN,NMAT,IPRCO,ISNA,IPGS,KORC,BTH,DTH
C Card with nodal data (1 to K) of the element (8I5)
C (NEL(NN,I),I=1,8)
   7    1    0    0    0    0           0.000000  0.000000
   8    5    6    7    3    2    1    4
C /14/ DATA ABOUT TIME FUNCTIONS (2I5)
C NTABFT,MAXTFT
   1  100
C a) Data about function in a table form (2I5)
C IBR,IMAX    (IMAX.LE.MAXTFT)
   1    2
C b) Values for argument - function (2F10.0)
C ((FN(I,IBR,J),I=1,2),J=1,IMAX)
0.0000e+00 1.0000e+0
1.0000e+04 1.0000e+0
C /15/ GENERAL DATA ABOUT LOADS (4I5,5X,5I5)
C NCF,NPP2,NPP3, NPGR,    NPLJ,NTEMP,NZADP,INDZS,ICERNE
   4    0    0    0        0    0    0    0    0
C /15-8/ DATA FOR PRESCRIBED DISPLACEMENTS (3I5,F10.0,I5)
C   N   IP   NC       FAK KORC
   3    2    1 5.000E+09    0
   4    2    1 5.000E+09    0
   7    2    1 5.000E+09    0
   8    2    1 5.000E+09    0
C /16/ FINAL CARD (A4)
STOP
```

**Прилог 5.** Скрипта за аутоматско покретање програма paks и извршење улазное датотеке kocka.dat, runpak.

paks < komande > provera

**Прилог 6.** Команде које се задају програму paks кроз скрипту runpak.

kocka.dat
kocka.lst
kocka.unv

kocka.neu

**Прилог 7.** Испис у датотеци provera ukoliko je proračun korektno izvršen.

```
   INPUT FILE NAME   <pak.dat>
 ind56=          0
   OUTPUT FILE NAME   <kocka.lst>
   OUTPUT GRAPHICS FILE NAME   <kocka.unv>
  FILE    kocka.unv          ALREADY EXISTS

            PRESS "ENTER" TO DELETE OR
            KEY   "N"     TO BYPASS
   OUTPUT NEU GRAPHICS FILE NAME   <kocka.neu>
  FILE    kocka.neu          ALREADY EXISTS

            PRESS "ENTER" TO DELETE OR
            KEY   "N"     TO BYPASS
*** INPUT DATA FOR ELEMENTS GROUP   1 /  1 ***
*** S T E P    N U M B E R     1 /   1,  TIME  1.00000D+00 ***
*** STIFFNESS MATRIX FOR GROUP   1 /  1, STEP   1 /   1, ITERATION  0 / 0 ***
*** SOLUTION EQUATIONS SYSTEM (1) ***
*** SOLUTION EQUATIONS SYSTEM (2) ***
*** STRESSES AND FORCES FOR GROUP   1 /  1, STEP   1 /   1, ITERATION  0 / 0 *
*** PRINT STRESSES FOR GROUP   1 /  1, STEP   1 /   1 ***
```

**Прилог 8.** Улазна датотека пограмског решења paks када је задата нелинеарност материјала.

```
C /1/ HEADING CARD (80A1)
C NASLOV
  Untitled
C /2/ FORMAT FOR INPUT DATA (free format)
C INDFOR
    2
C /3/ BASIC DATA FOR THE PROBLEM (4I5,5X.I5)
C NP,NGELEM,NMATM,NPER,NKRT,NCRACK,NCXFEM,IGBM
    8    1    1    1
C /4/ BASIC DATA FOR THE PROBLEM (6I2,3X,3I5)
C (IOPGL(I),I=1,6),NDIN,ISOPS,ICVEL
 0 0 0 1 1 1       0    0    1
C /5/ DATA FOR POST-PROCESSING (10I5)
C NULAZ,NBLPR,NBLGR,ISTKO,NCVPR,ISTEM,ISTVN,ISTSI,ISTDE,ISTNA
    0    0    1    0    0    0    0    0    0    0    0
C /6/ GROUP OF CARDS WITH DATA FOR BLOCKS (5I5)
C NBL,(NPRGR(NBL,I),I=1,4)
    1    1    1 9000    0
C /7/ DATA FOR RESTART (I5,10X,2I5,2F10.0)
C IREST,       NMODS,ICCGG,    TOLG,    ALFAG
    0            0    0  0.000000  0.000000         0
C /8/ GROUP OF CARDS WITH DATA FOR TIME STEPS (2I5,F10.0)
C   I,NKDT(I),DTDT(I)    (I=1,NPER)
    1   30  1.000000
C /9/ DATA FOR EQULIBRIUM ITERATIONS (5I5,4F10.0,I5)
C METOD,MAXIT,KONVE,KONVS,KONVM,TOLE,    TOLS,     TOLM,
TOLA,NBRCR
    4   50    1    0    0  0.000000  0.001000  0.000000  0.000000    2
C /10/ GROUP OF CARDS FOR NODAL POINT DATA (I5,A1,6I2,2X,3F10.0,2I5)
(K=1,NP)
C  N,CH,(ID(N,I),I=1,6),   (CORD(N,J),J=1,3),      KORC,LJ(N)
    1  1 1 0 0 0 0     0.00000   0.00000   1.00000    0    0
    2  0 1 0 0 0 0     1.00000   0.00000   1.00000    0    0
    3  0 0 0 0 0 0     1.00000   1.00000   1.00000    0    0
    4  1 0 0 0 0 0     0.00000   1.00000   1.00000    0    0
    5  0 1 1 0 0 0     1.00000   0.00000   0.00000    0    0
    6  1 1 1 0 0 0     0.00000   0.00000   0.00000    0    0
    7  1 0 1 0 0 0     0.00000   1.00000   0.00000    0    0
    8  0 0 1 0 0 0     1.00000   1.00000   0.00000    0    0
C /11/ GENERAL DATA FOR MATERIAL MODELS (3I5)
C (MODEL(I,K),I=1,3)    (K=1,NMATM)
    6    1   20
C /12/ DATA FOR EACH MATERIAL (2I5,F10.0)
C MOD  MAT     GUST
    6    10.0000e+00
C /12-6/ MATERIAL MODEL 6 (MISES ELASTIC-PLASTIC WITH MIXED
HARDENING)
```

```
C Number of points for definition of function strain-stress (I5)
C NTFUN
   1
C Modulus of elasticity and Poissons coefficient (2F10.0)
C     E,    V
2.000e+0113.000e-001
C Hardening function - Ramberg-Osgood (2F10.0)
C  TAUY,   Cy
150000000.150000000.
C Hardening function Ramberg-Osgood  (2F10.0)
C    AN,   EM
.9000000001.000e+000
C /13/ ELEMENT GROUP DATA (7I5,3F10.0)    (I=1,NGELEM)
C NETIP,NE,IATYP,NMODM,INDBTH,INDDTH,INDKOV, COEF1, COEF2, COEF3
   3    1    1    6    0    0    0    0 0.000000  0.000000  0.000000
C /13-3/ DATA FOR 3D ISOPARAMETRIC ELEMENTS
C Card with basic data about 3D elements (3I5,5X,F10.0,35X,I5
C NGAUSX,NGAUSY,NGARSZ,BETA,IALFA
   2    2    2       0.000000                                    0
C Card with data about the current element (6I5,10X,2F10.0)
C NN,NMAT,IPRCO,ISNA,IPGS,KORC,BTH,DTH
C Card with nodal data (1 to K) of the element (8I5)
C (NEL(NN,I),I=1,8)
   7    1    0    0    0    0          0.000000  0.000000
   8    5    6    7    3    2    1    4

C /14/ DATA ABOUT TIME FUNCTIONS (2I5)
C NTABFT,MAXTFT
   1  100
C a) Data about function in a table form (2I5)
C IBR,IMAX    (IMAX.LE.MAXTFT)
   1    2
C b) Values for argument - function (2F10.0)
C ((FN(I,IBR,J),I=1,2),J=1,IMAX)
0.0000e+00 0.0000e+0
5.0000e+01 1.0000e+1
C /15/ GENERAL DATA ABOUT LOADS (4I5,5X,5I5)
C NCF,NPP2,NPP3, NPGR,    NPLJ,NTEMP,NZADP,INDZS,ICERNE
   4    0    0    0        0    0    0    0    0
C /15-1/ CONCENTRATED LOADS DATA (3I5,F10.0,I5,F10.0)
C    N   IP   NC   FAK     KORC    FPOM
   3    2    1 5.0000e+7    0  0.000000
   4    2    1 5.0000e+7    0  0.000000
   7    2    1 5.0000e+7    0  0.000000
   8    2    1 5.0000e+7    0  0.000000
C /16/ FINAL CARD (A4)
STOP
```