Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

Visokošolski program Gradbeništvo,
Smer operativno gradbeništvo

Kandidat:
Andrej Kovačič

Avtomatizacija edometrske preiskave

Diplomska naloga št.: 302

Mentor:
izr. prof. dr. Janko Logar

Somentor:
asist. dr. Boštjan Pulko

Ljubljana, 25. 2. 2008

BIBLIOGRAFSKO-DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: 624.13+624.15(043.2)

Avtor: Andrej Kovačič

Mentor: doc. dr. Janko Logar, univ.dipl.ing.grad.

Somentor: asist. dr. Boštjan Pulko univ. dipl. ing.grad.

Naslov: Avtomatizacija edometrske meritve

Obseg in oprema: 71 str., 33 sl., 17 en., 2 pril.

Ključne besede: mehanika tal, temeljenje, konsolidacija, edometrska preiskava, objektno programiranje

Izvleček

Edometrska preiskava je pogosto izvajana preiskava v geomehanskih laboratorijih. Njeni rezultati imajo velik vpliv pri projektiranju konstrukcij. Rezultati edometrske preiskave so podatki za napoved posedkov pod objektom in časovnega poteka posedanja. Čeprav je razvoj merilne tehnike edometrsko preiskavo zelo avtomatiziral, je analiza stanja v laboratoriju KMTAL, FGG pokazala možnosti za izboljšave. V tej diplomski nalogi je bil posodobljen obstoječi avtomatiziran sistem, pri katerem na novo izdelan računalniški program omogoča nadzor nad potekom meritve v realnem času z grafičnim vmesnikom. Omogoča tudi kombinacijo ročne in avtomatizirane meritve, prostorsko ločenost kontrolne in merilne računalniške aplikacije, ter zapis rezultatov v obliki, primerni za nadaljnjo obdelavo. Pri razvoju programske opreme so bile uporabljene sodobne računalniške tehnologije: objektno programiranje, povezava programov na daljavo, zvočna komunikacija in večnitno programiranje. Izdelava končnega programa je potekala s pomočjo prototipa. Tako so bile že v fazi razvoja odpravljene vse bistvene pomanjkljivosti zasnove sistema. Poleg tega je prototipska aplikacija porodila tudi nove ideje, ki so uporabljene v končni aplikaciji. V roku enega leta je bilo z novim sistemom uspešno in brez težav izmerjenih nekaj 10 edometrskih meritev.

BIBLIOGRAPHIC-DOCUMENTALISTIC INFORMATION

UDK: 624.13+624.15(043.2)

Author: Andrej Kovačič

Mentor: Assistant Professor, PhD. Janko Logar, Civil Engineer.

Co-mentor: Assistant, PhD. Boštjan Pulko, Civil Engineer.

Title: Oedometer test automation

Volume and equipment: 79 p., 33 fig., 17 eq., 2 app.

Keywords: soil mechanics, foundation engineering, consolidation, oedometer test, object oriented programming

Abstract

In geomechanical laboratories the oedometer test is a very often executed investigation. The results have a significant influence on the constructions design. Oedometer laboratory test results are the data for the prediction of the settlements under construction and their time development. Though the oedometer test has been, due to the development of the measuring techniques, highly automated, the analysis of the state of the art in the laboratory KMTAL, FGG, has shown some possibilities of improvement. In the present work the existing automated system has been updated. The new developed computer program enables a interactive graphic control over the course of the measurements in real time. It enables combination of the manual and the automated measurement, space separated control and measurement computer applications and output results in the format, suitable for further processing, as well. In the computer program development several up to date software engineering technologies has been applied: object oriented programming, remotely distributed programs and multithreaded programming. The final program was designed and elaborated by means of a prototype. In this manner, all essential deficiencies of the design of the system have been eliminated. Besides, the prototype application has initiated some new ideas, which were included in the final application. In the period of one year some 10 oedometer tests has been successfully performed by the new system.

ZAHVALA

Lepo se zahvaljujem:

- mentorju, doc. dr. Janku Logarju za mentorstvo in koristne napotke pri izdelavi diplomske naloge,

- laborantom Miranu Mercu, Vanji Selan in Mateju Mačku za nasvete, pripombe in potrpežljivost pri uvajanju sistema v praktično uporabo,

- svojim staršem za moralno in finančno podporo v času študija,

KAZALO VSEBINE

KAZALO SLIK

1 UVOD

1.1 Konsolidacija zemljine

Konsolidacija zemljine je geološki proces, pri katerem pride do zmanjševanja volumna zemljine izpostavljene zunanjim pritiskom, ki povzročijo, da se pore med delci zemljine pričnejo zmanjševati. Če je zemljina nasičena z vodo, se slednja pod vplivom zunanjih pritiskov prične izcejati, delci zemljine pa prevzemajo vnesene napetosti. Vodo med porami zadržujeta kapilarni tlak in kemični procesi.

Velikost in potek konsolidacije zemljine je možno izmeriti in napovedati z različnimi metodami. Po klasični metodi K. Terzaghija vzorec zemljine izpostavimo enoosni tlačni napetosti, ki se stopenjsko povečuje. Meri se časovni potek spreminjanja višine vzorca.

V praksi je posledica procesa konsolidacije zemljine posedanje objektov, zato je potrebno njene vplive pri projektiranju skrbno upoštevati. Velikokrat se zgodi, da je en del objekta temeljen na manj stisljivi podlagi kot drugi del. V tem primeru bo posedanje objekta neenakomerno, kar je še posebno neugodno, saj se zaradi velike lastne teže na konstrukcijo vnesejo velike obremenitve.

1.2 Pomen edometrske preiskave pri projektiranju objektov

V fazi projektiranja je torej potrebno določiti optimalno razmerje med poboljšanjem tal in prilagoditvijo objekta posedkom z upoštevanjem finančnih, uporabnostnih in drugih kriterijev. Rezultati edometrske meritve so vhodni podatki za napoved časovnega poteka in končnega posedanja objekta.

Zaradi izredne geološke pestrosti pri nas je edometrska preiskava ena izmed najpogosteje izvajanih preiskav v geomehanskih laboratorijih.

Razvoj merilne tehnike je do današnjih dni edometrsko preiskavo skoraj povsem avtomatiziral, tako da se izvajanja preiskave s štoparico v eni in s svinčnikom v drugi roki poslužujejo le redkokje oz. v izjemnih primerih. Razvoj je prinesel napredek tudi pri vseh ostalih zvrsteh preiskav, kar je zmogljivosti geomehanskega laboratorija močno povečalo.

Povečala se je količina podatkov, ki jih je potrebno obdelati in tudi arhivirati. Razvili so se novi standardi shranjevanja podatkov in nove programirne tehnike, ki omogočajo lažje in učinkovitejše razvijanje programskih sistemov.

1.3 Avtomatizacija edometrske preiskave

Za izvedbo avtomatizirane edometrske preiskave obstaja danes več kvalitetnih in zmogljivih komercialnih merilnih sistemov različnih proizvajalcev. Ob hitrem pregledu ponudb na svetovnem spletu vidimo, da imajo vsi visoko natančnost, robustnost in zanesljivost na eni strani, ter visoko ceno, nekompatibilnost in zaprtost navzven na drugi. Proizvajalec merilne opreme je večinoma tudi izdelovalec pripadajoče programske opreme, ki je praviloma toga, dobljene rezultate pa je težko arhivirati in prenašati med različnimi programi za končno obdelavo.

Pri razvoju programske opreme je zelo uveljavljen način izdelave programske opreme sestavljanje programa iz programskih komponent, ki še danes velja kot zelo učinkovit in zmogljiv pristop. Vendar je lahko komponentna izdelava pri obsežnih tehničnih programih kompleksna in zahteva posredno več razvojnega časa. V takih primerih je primerneje izdelati več manjših samostojnih programov, ki komunicirajo med seboj.

1.4 Cilj diplomske naloge

Cilj diplomske naloge je izdelava programskega sistema za izvajanje edometrske preiskave. Pri razvoju sem se trudil uporabiti najsodobnejše koncepte programiranja in razviti enostavno, robustno in fleksibilno edometrsko aplikacijo. Sistem sestavljata dva programa z imenom *Edometer V2* in *Memo Upravljalnik*. Program *Edometer V2* je namenjen izvedbi edometrske meritve, *Memo Upravljalnik* pa nadzoru delovanja strojne opreme in distribuciji merjenih vrednosti.

Oba programa sta naslednika prototipskega programa *MHT Edometer*, izdelanega v obdobju obveznega praktičnega usposabljanja.

V naslednjih poglavjih so predstavljene teoretične osnove edometrske meritve, potek edometrske meritve v laboratoriju KMTAL, opis izvedbe merilnega sistema s programoma *Edometer V2* in *Memo Upravljalnik*, prikaz uporabe programov in zaključek.

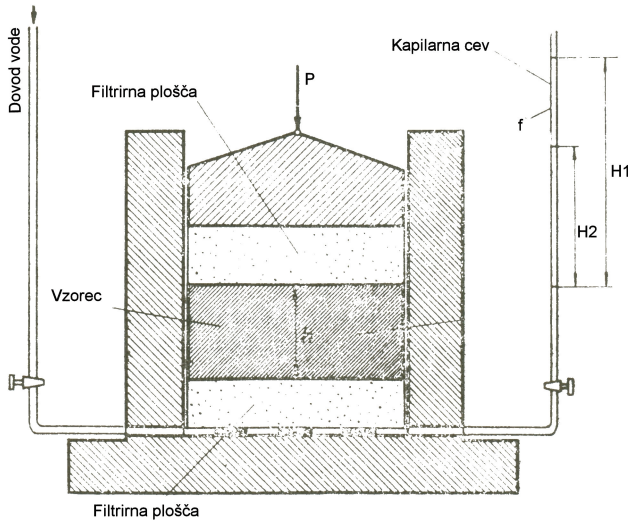
2 TEORETIČNE OSNOVE EDOMETRSKE MERITVE

2.1 Uvod

Za določitev posedkov pod objektom moramo poznati specifično deformabilnost zemljine oz. edometrski modul. Edometrski modul določa deformacijo sloja zemljine pod dano obtežbo po končani konsolidaciji.

K. Terzaghi (Terzaghi, 1943) je uvedel preiskavo deformabilnosti zemljin, pri kateri je vzorec vtisnjen v tog kovinski valj premera od 7 do 10 cm, ki preprečuje deformacije v bočni smeri. Višina vzorca je od 1 do 4 cm. Vzorec je preko bata in običajno mehanskega sistema prenosa sile izpostavljen konstantni napetosti. Z merilno urico merimo deformiranje vzorca tekom delovanja napetosti. Obtežbo na vzorec vnašamo po stopnjah, posamezno napetostno ali bremensko stopnjo pa opazujemo tako dolgo, da postane hitrost deformacije dovolj majhna, oz. da lahko iz časovnega poteka deformacije dovolj zanesljivo sklepamo o njenem nadaljnjem poteku.

Tekom preiskave moramo biti posebej pozorni, da se vzorec ne izsušuje in ga v ta namen preplavimo, oz. poskrbimo za nasičeno vlažno atmosfero okoli njega.

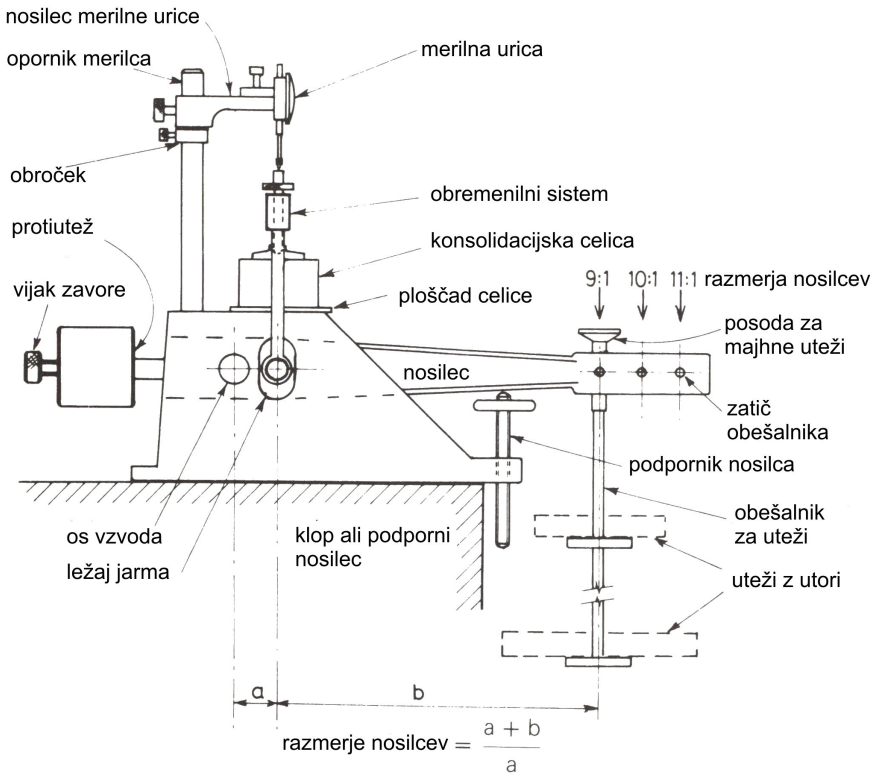


Slika 1: Shematski prikaz edometrskega aparata

Obstaja več vrst edometrskih aparatov. Ločijo se glede načina obremenjevanja vzorca, ki je lahko izvedeno bodisi mehansko preko vzvoda bodisi hidravlično bodisi pnevmatsko.

Mehanski način je enostaven in razmeroma zanesljiv, saj poteka večinoma ročno z nalaganjem uteži na vzvod aparata. Žal pa je vzorec v kratkem času težko povsem zvezno

obremeniti oz. razbremeniti. Zahtevano sodelovanje človeka tudi onemogoča avtomatizacijo preiskave.



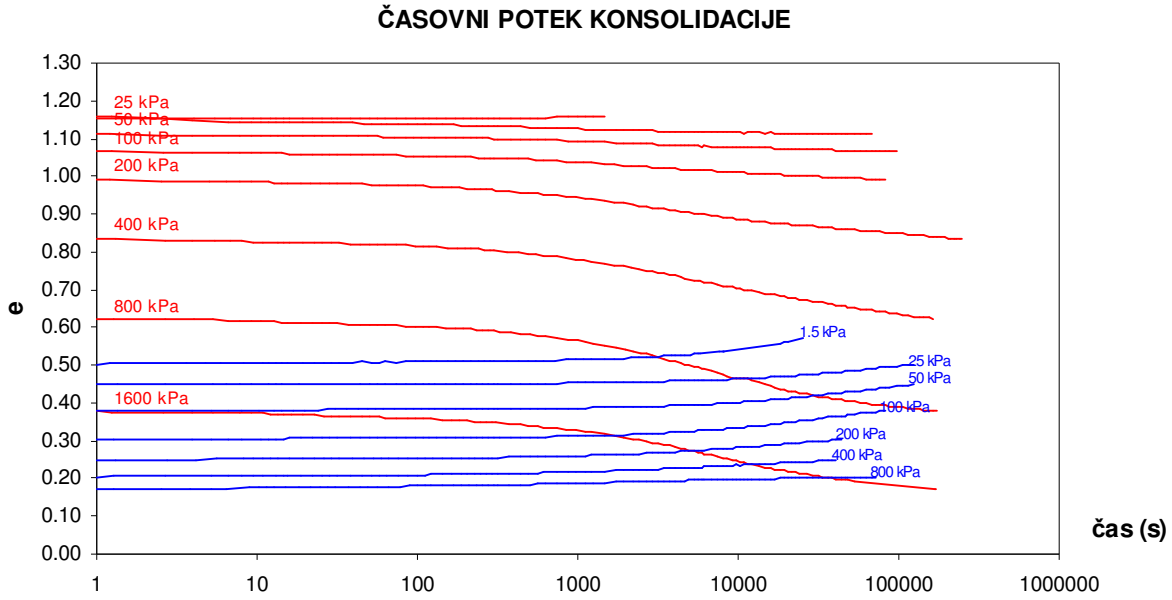
Slika 2: Prikaz mehanskega edometrskega aparata

Pri hidravličnem obremenjevanju vzorec obremenjuje hidravlična batnica, pri pnevmatskem pa balon. Oba načina zahtevata precizno regulacijsko tehniko, omogočata zvezno in natančno obremenjevanje in popolno avtomatizacijo edometrske preiskave.

Pri nas predpisuje izvedbo edometrskega preizkusa, priporočila in zahteve glede merilnega instrumentarija standard (ISO/TS 17892-5:2004). Velikost obremenjevanja in trajanje njenega delovanja v posamezni bremenski stopnji določajo naslednji kriteriji: trenutni geološki tlaki zemljine, geološki tlaki v preteklosti, končni tlaki pod objektom, tlaki pod objektom v fazah gradnje in priporočila standarda glede na vrsto zemljine.

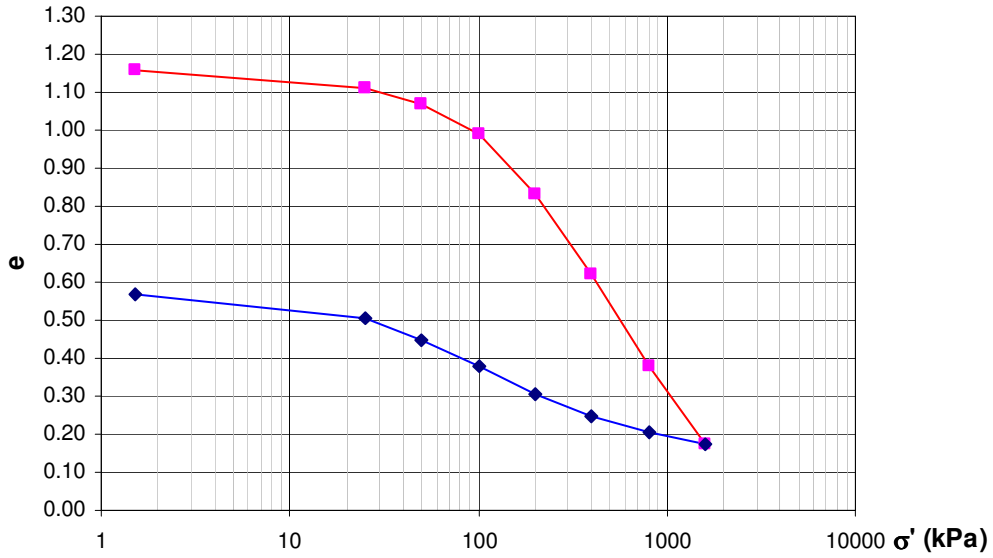
Rezultati edometrske preiskave so običajno prikazani v treh različnih diagramih. Ti so: diagram časovnega poteka konsolidacije, diagram krivulje stisljivosti in diagram edometrskega modula.

V diagramu časovnega poteka konsolidacije kot neodvisna spremenljivka nastopa *čas od začetka bremenske stopnje*, kot odvisna pa *višina vzorca oz. količnik por*.



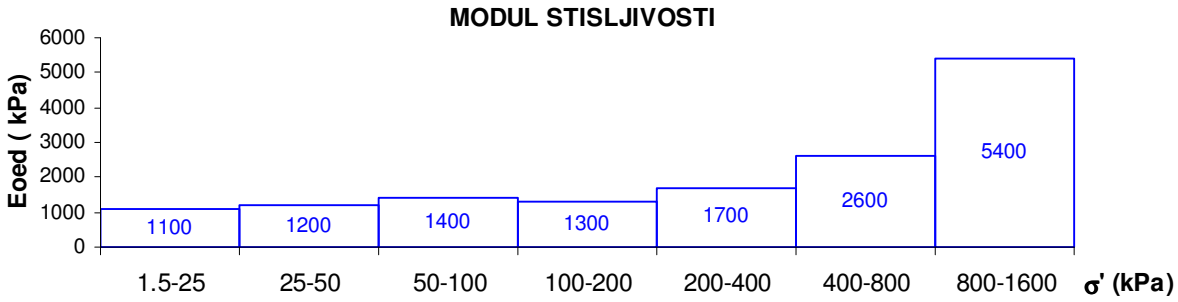
Slika 3: Prikaz časovnega poteka konsolidacije

V diagramu krivulje stisljivosti kot neodvisna spremenljivka *nastopa obremenitev oz. napetost vzorca*, odvisna pa *količnik por vzorca*.



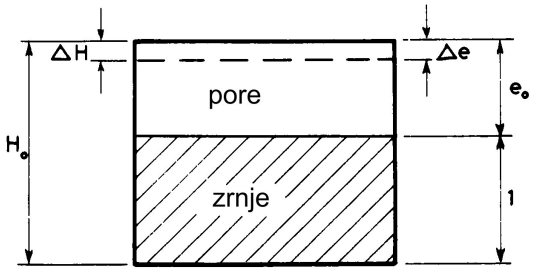
Slika 4: Prikaz krivulje stisljivosti

V diagramu edometrskega modula kot neodvisna spremenljivka nastopa *vertikalna* napetost *bremenske stopnje*, kot odvisna pa *vrednost edometrskega modula*.



Slika 5: Prikaz edometrskih modulov

Količnik por predstavlja razmerje med volumnom praznin v zemljini in volumnom zrnja.



Slika 6: Prikaz pomena količnika por

Količnik por danega vzorca lahko izračunamo šele po koncu meritve, saj dobimo volumen stisnjenega zrnja s tehtanjem povsem posušenega vzorca in z oceno oz. z meritvijo specifične teže zrnja.

Če želimo prikazovati diagrame časovnega poteka konsolidacije in krivulje stisljivosti med izvajanjem preiskave, lahko odvisno spremenljivko *količnik por* nadomestimo z *višino vzorca oz. relativno višino vzorca*, ki jo dobimo med merjenjem. Tako dobljeni diagrami imajo enake značilnosti kot zgoraj navedeni, saj so samo premaknjeni in razširjeni v vertikalni smeri.

V zgoraj navedenih diagramih lahko s pomočjo grafičnih metod izračunamo še dodatne parametre materiala. V nadaljevanju bom najprej predstavil uporabljen računski model za osnovni izračun konsolidacijskih parametrov, nato sledi predstavitev dodatnih grafičnih metod.

2.2 Računski model edometrske preiskave

Računski model predstavljen v tem poglavju obravnava računske postopke potrebne za izdelavo načrtovane računalniško podprte avomatizirane edometrske meritve, ki je centralni cilj te diplomske naloge. Ključni izmerjeni in posredno izračunani podatki, ki omogočajo zanesljiv nadzor nad potekom meritve, so diskretni (po točkah in bremenskih stopnjah).

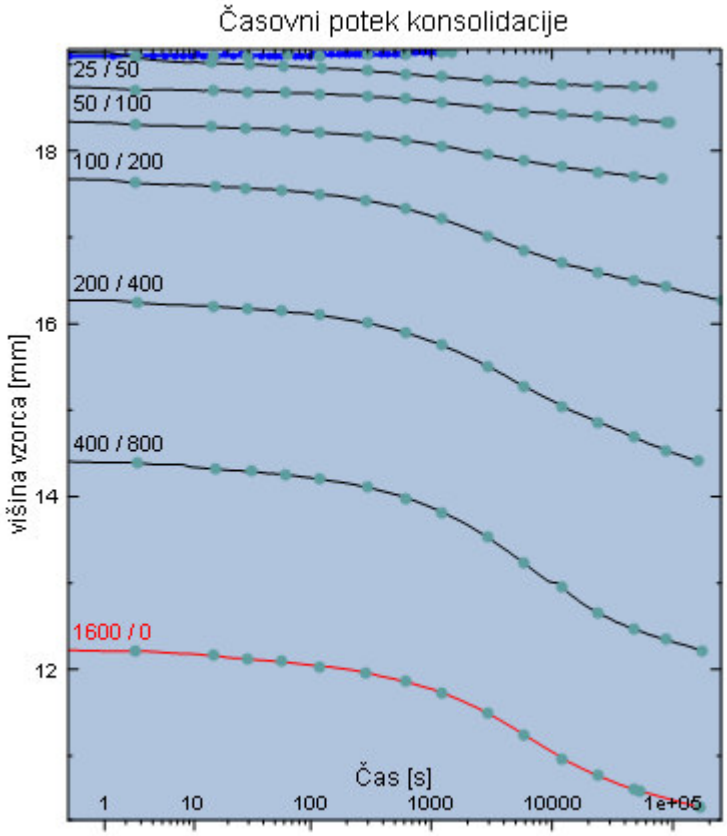
2.2.1 Uporabljeni simboli:

$H_{i,j}$	višina vzorca pri meritvi ob času j v bremenski stopnji i
H_0	višina obroča aparata oz. začetna višina vzorca
H_i	višina ob koncu konsolidacije (opazovanja) bremenske stopnje i
H_s	višina zrnja vzorca
V_i	volumen vzorca ob koncu konsolidacije (opazovanja) bremenske stopnje i
V_s	volumen zrnja vzorca
$O_{i,j}$	odčitek na merilni urici ob časovnem koraku j v bremenski stopnji i
O_i	odčitek ob koncu konsolidacije (opazovanja) bremenske stopnje i
$O_{1,1}$	prvi odčitek prve bremenske stopnje
$T_{i,j}$	čas meritve ob časovnem koraku j v bremenski stopnji i
$T_{i,1}$	čas prve meritve v bremenski stopnji i
$t_{i,j}$	čas od začetka bremenske stopnje do meritve ob časovnem koraku j v bremenski stopnji i (relativni čas)
$e_{i,j}$	delež por pri meritvi j v bremenski stopnji i
e_i	delež por pri zadnji meritvi v bremenski stopnji i
σ_i	napetost bremenske stopnje i
ε_i	vzdolžna deformacija vzorca pri zadnji meritvi v bremenski stopnji i
Mv_i	edometrski modul pri zadnji meritvi v bremenski stopnji i
M_i	masa obremenilne uteži bremenske stopnje i
D	premer obroča aparata (preizkušanca)
A	presek vzorca
m_d	masa suhega vzorca
ρ_s	gostota zrnja zemljine
g	težnostni pospešek
η	prenosno razmerje obtežnega mehanizma aparata

c_v	koeficient konsolidacije
k	koeficient prepustnosti
γ_w	specifična teža vode
T_v	časovni faktor konsolidacije
U_v	stopnja oz. delež konsolidacije
d	drenažna višina
OCR	prekonsolidacijsko razmerje
p_c	največji efektivni tlak, pri katerem je bila zemljina konsolidirana v preteklosti
p_0	trenutni efektivni tlak v zemljini

2.2.2 Model edometrske preiskave

Ko je vzorec pripravljen, vgrajen v aparat in je preiskava v teku, poteka merjenje višine vzorca ob diskretnih časih. Z izmerjenimi podatki lahko tekom merjenja rišemo diagram časovnega poteka konsolidacije, ki nudi zelo pregleden prikaz poteka preiskave. Kakršno koli napako v obremenjevanju lahko takoj registriramo in preiskavo po potrebi ponovimo.



Slika 7: Diagram časovnega poteka konsolidacije programa Edometer V2

Točke za izris diagrama na sliki 7 izračunamo po naslednjih formulah.

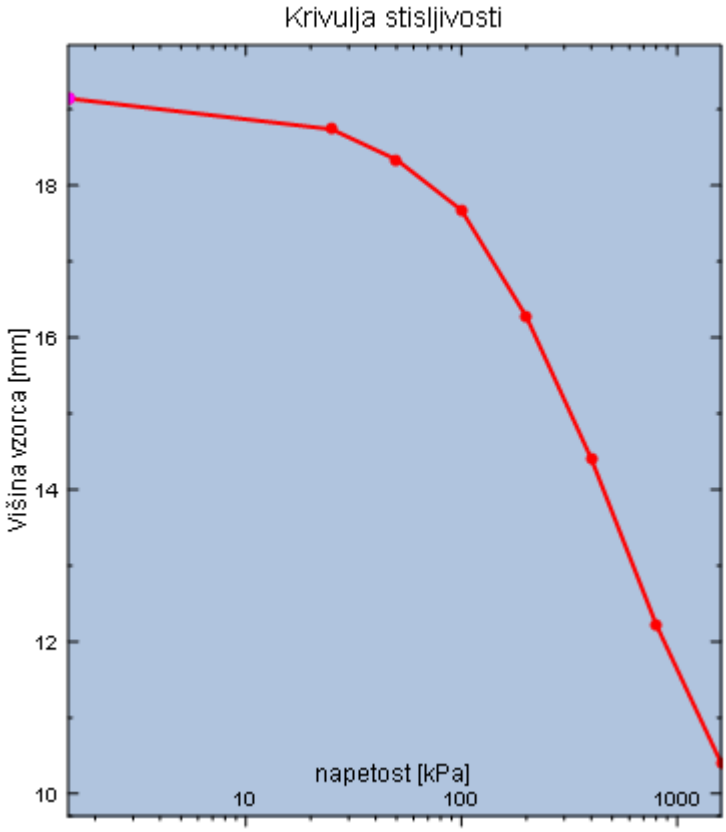
$$t_{i,j} = T_{i,j} - T_{i,1} \qquad \text{Enačba 1}$$

$$H_{i,j} = H_0 - \left(O_{i,j} - O_{1,1}\right) \qquad \text{Enačba 2}$$

Po enačbi 1 izračunamo *čas od začetka bremenske stopnje*, v nadaljevanju *relativni čas*.

Po enačbi 2 izračunamo *diskretne višine vzorcev*. Formula velja le v primeru, da se odčitek na urici pri stiskanju vzorca povečuje.

Ko je meritev v teku in smo končali že nekaj bremenskih stopenj, nas običajno zanima krivulja stisljivosti. Njena oblika veliko pove o tem, kako je bil vzorec obremenjevan v preteklosti, kar sledi v naslednjem poglavju.



Slika 8: Diagram krivulje stisljivosti $H_i(\sigma_i)$ programa Edometer V2

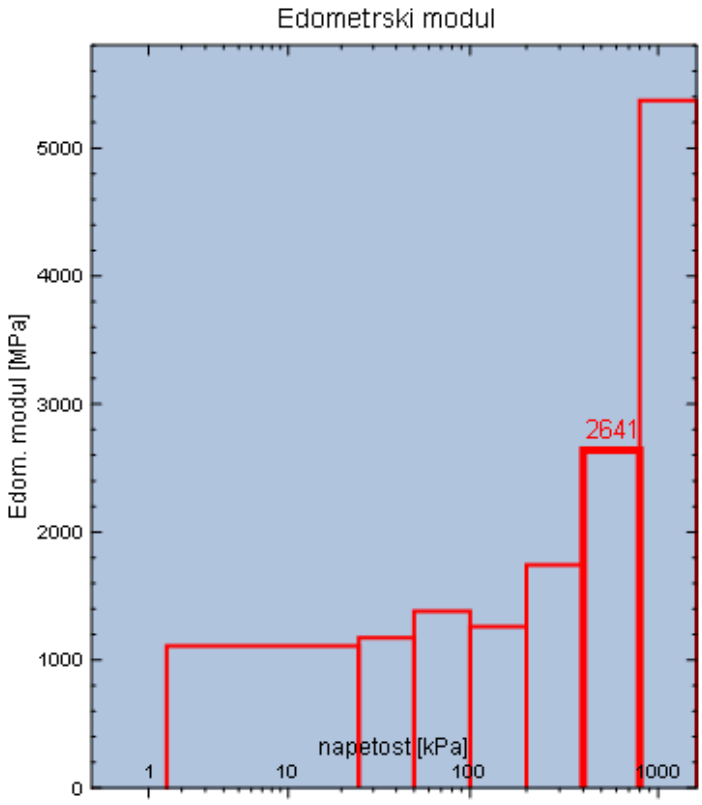
Točke za izris diagrama krivulje stisljivosti izračunamo po naslednjih formulah.

$$\sigma_i = \frac{M_i \cdot g \cdot \eta \cdot 4}{\pi \cdot D^2} \qquad \text{Enačba 3}$$

$$H_i = H_0 - \left(O_i - O_{1,1}\right) \qquad \text{Enačba 4}$$

Enačba 4 velja, če se odčitek na urici pri stiskanju vzorca povečuje. Po enačbi 3 izračunamo *napetost v vzorcu*, po enačbi 4 pa *višine vzorcev ob koncu konsolidacijske stopnje*.

Z merjenimi podatki, s katerimi smo lahko izrisali diagrama poteka konsolidacije, lahko izrišemo tudi diagram edometrskih modulov. Edometrski modul določa, za kakšen delež se bo zemljina stisnila pod dano obtežbo po končani konsolidaciji. Edometrski modul ni konstanten, ampak je odvisen od napetosti in deleža oz. stopnje konsolidacije.



Slika 9: Diagram edometrskega modula programa Edometer V2

Točke za izris diagrama edometrskega modula izračunamo po naslednjih formulah.

$$Mv_i = \frac{\sigma_i - \sigma_{i-1}}{\varepsilon_i - \varepsilon_{i-1}}$$ Enačba 5

$$\varepsilon_i = \frac{H_0 - H_i}{H_0}$$ Enačba 6

Če združimo enačbe 3, 4, 5 in 6 za izračun edometrskega modula, dobimo naslednjo formulo.

$$Mv_i = \frac{(M_i - M_{i-1}) \cdot H_0 \cdot g \cdot \eta \cdot 4}{(O_i - O_{i-1}) \cdot \pi \cdot D^2}$$ Enačba 7

Ko preiskavo končamo, vzorec stehtamo, posušimo in ponovno stehtamo. Tako dobimo tudi podatke za izračun količnika por. Količnik por predstavlja razmerje med volumnom praznin v vzorcu in volumnom zrnja.

$$e_i = \frac{V_i - V_s}{V_s}$$ Enačba 8

Ker je presek vzorca konstanten, velja:

$$e_i = \frac{H_i - H_s}{H_s}$$ Enačba 9

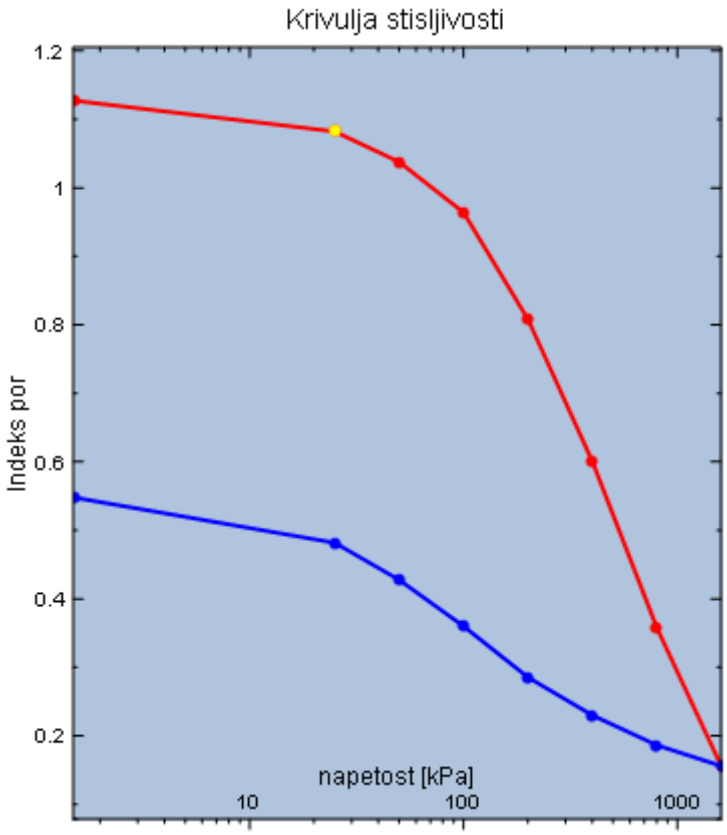
Višino zrnja vzorca določimo na podlagi mase posušenega vzorca po končani preiskavi in na podlagi ocenjene oz. izmerjene gostote zrnja.

$$H_s = \frac{m_d}{\rho_s \cdot A} \qquad \text{Enačba 10}$$

Ko združimo enačbe 4, 9 in 10, dobimo končno formulo za izračun deleža por ob koncu bremenske stopnje.

$$e_i = \frac{H_0 - (O_i - O_{1,1})}{\frac{m_d}{\rho_s \cdot A}} - 1 \qquad \text{Enačba 11}$$

Sedaj lahko *višino vzorca* v krivulji stisljivosti nadomestimo s *količnikom por*. Dobimo naslednji diagram.



Slika 10: Diagram krivulje stisljivosti $e_i(\sigma_i)$ programa Edometer V2

Če primerjamo diagrama na sliki 8 in sliki 10, vidimo, da je oblika obremenilne krivulje $H_i(\sigma_i)$ in $e_i(\sigma_i)$ (rdeča črta) povsem enaka v obeh diagramih. Diagram na sliki 10 ima v primerjavi z diagramom na sliki 8 prikazane tudi razbremenilne stopnje (modra črta), saj ob času izrisa diagrama na sliki 8 razbremenilne stopnje še niso bile izvedene.

Diagrama časovnega poteka in krivulje stisljivosti v programu *Edometer V2* sta izvedena interaktivno in uporabniku omogačata hiter grafični izračun dodatnih parametrov zemljine. Teoretično ozadje in računski model dodatnih postopkov je predstavljen v naslednjem poglavju.

2.2.3 Računski model dodatnih grafičnih postopkov pri edometrski preiskavi

Taylorjev postopek za izračun konca primarne konsolidacije

Postopek določanja konca primarne konsolidacije je leta 1942 predstavil Taylor (Taylor, 1942). Postopek temelji na teoretični odvisnosti stopnje konsolidacije U_v od časovnega faktorja konsolidacije T_v.

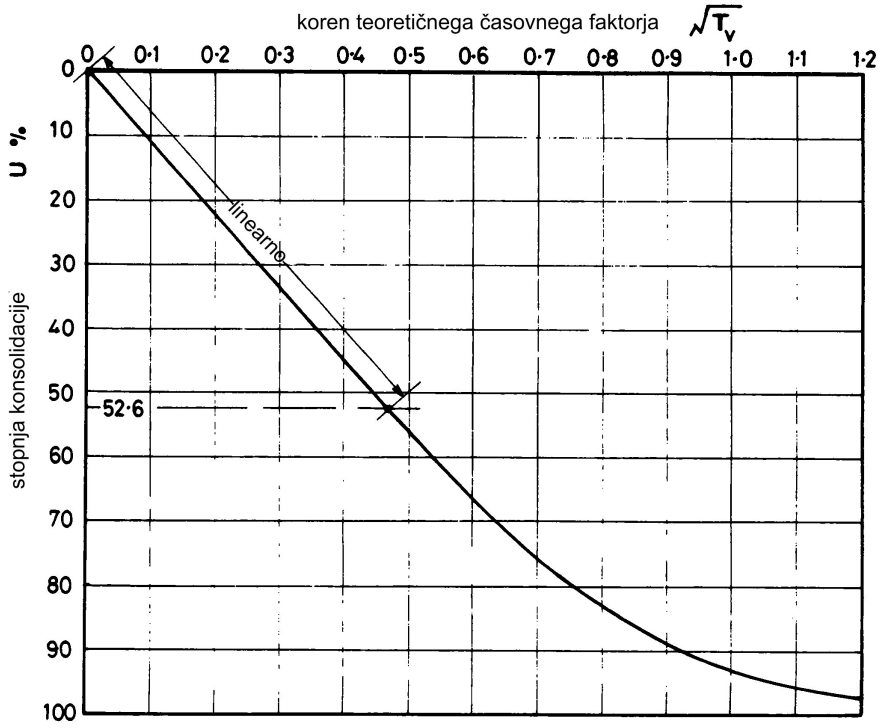
Pri izpeljavi enačb konsolidacijske teorije za časovni potek konsolidacije brezkrajnega sloja pod konstantno obtežbo pridemo do naslednje enačbe, ki ponazarja zvezo med časom in stopnjo konsolidacije.

$$U_v = 1 - \frac{8}{\pi^2} \cdot \sum_{\substack{m=1 \\ liho}}^{+\infty} \frac{e^{-m^2\pi^2 T_v / 4}}{m^2} \qquad \text{Enačba 12}$$

Enačba 12 predstavlja brezdimenzijsko odvisnost med stopnjo konsolidacije in časovnim faktorjem. Aproksimirano jo lahko zapišemo v enačbah 13 in 14.

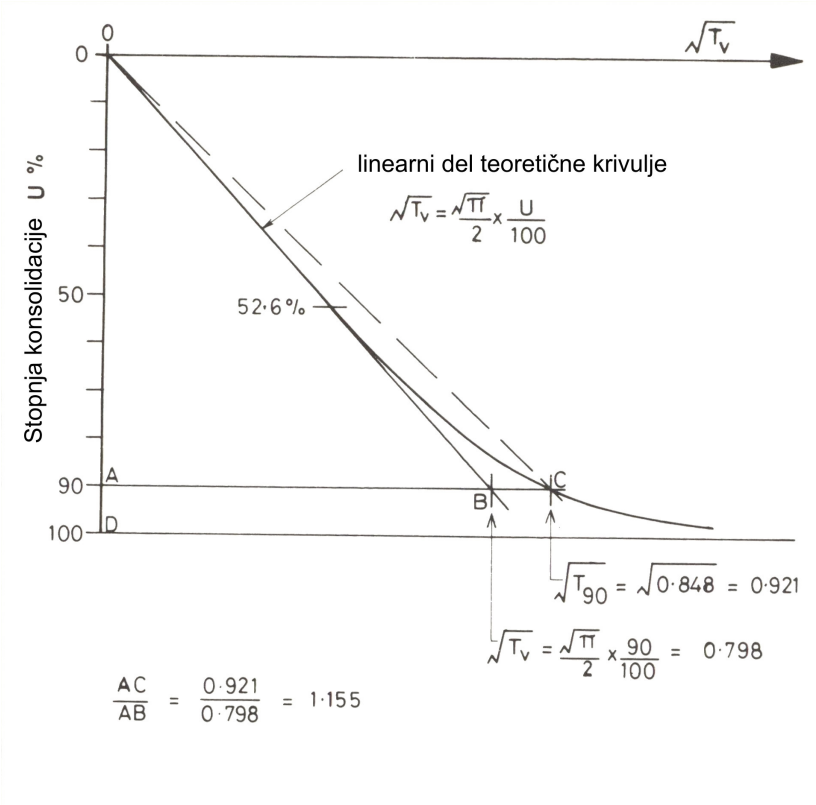
$$U_v(T_v) = 2 \cdot \sqrt{\frac{T_v}{\pi}} \rightarrow T_v < 0{,}217 \qquad \text{Enačba 13}$$

$$U_v(T_v) = 1 - \frac{8}{\pi^2} \cdot e^{-\frac{\pi^2}{4} \cdot T_v} \rightarrow T_v \geq 0{,}217 \qquad \text{Enačba 14}$$



Slika 11: Odvisnost $U_v(T_v)$ prikazana pri korenskem merilu abscise

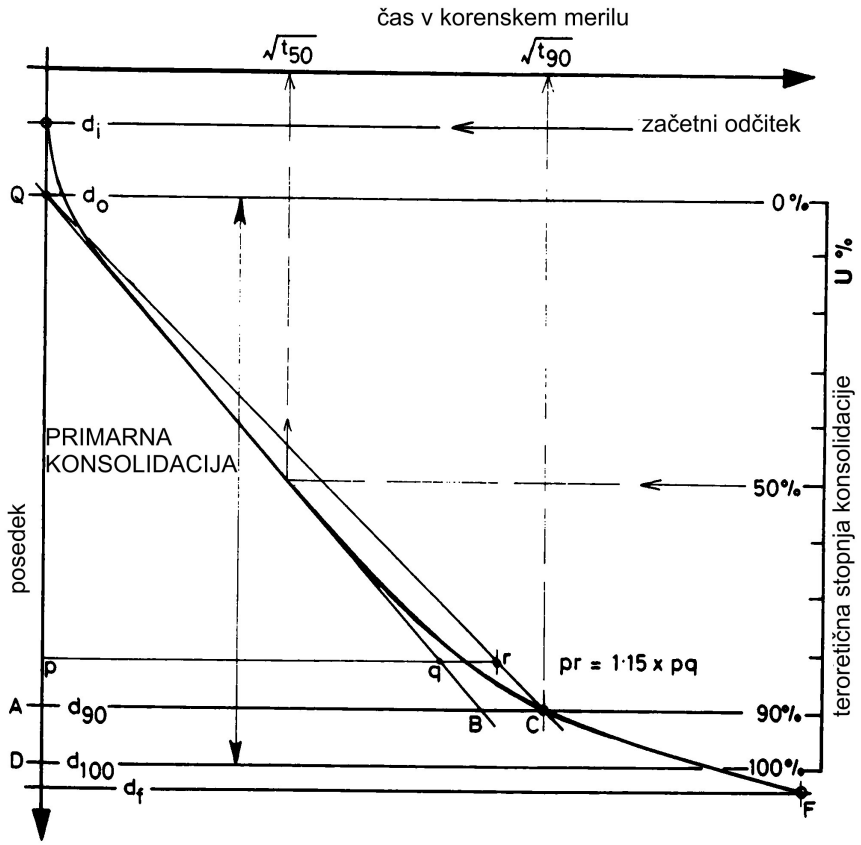
Slika 11 predstavlja aproksimirano odvisnost $U_v(T_v)$ na podlagi enačb 13 in 14 v korenskem merilu abscise. Vidimo, da je do stopnje konsolidacije 0,5, zveza med korenom časovnega faktorja in stopnjo konsolidacije linearna.



Slika 12: Teoretičen prikaz Taylorjevega postopka

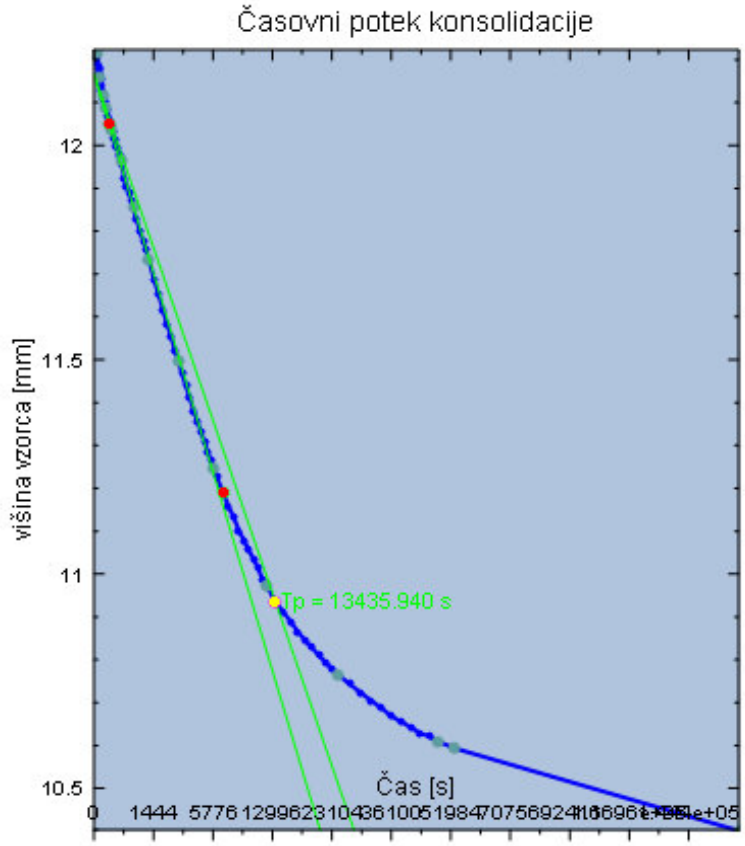
Konsolidacija vzorca v edometrskem aparatu ima 3 faze. Prva faza je faza začetne konsolidacije, ki se zgodi skoraj hipno ob vnosu obtežbe na vzorec. Zgodi se kot posledica elastične deformacije materiala ter zaradi stiskanja in prerazporeditve majhnih mehurčkov zraka ujetih v prostoru med porami. Dodatno obtežbo prevzame voda v prostoru med porami, ki se v fazi primarne konsolidacije izceja iz vzorca, zrnje pa postopoma prevzema dodatno obtežbo.

Med koncem 1. faze in do sredine 2. faze konsolidacije je realna krivulja konsolidacije v korenskem merilu časa skoraj povsem linearna in se lepo ujame s teoretičnimi predpostavkami.



Slika 13: Prikaz faz konsolidacije in uporaba Taylorjevega postopka

Ko so porni tlaki zaradi dodatne obtežbe enaki 0, govorimo o nastopu sekundarne konsolidacije. Povzročijo jo prerazporeditve zrn materiala pri konstantni efektivni napetosti. Pravzaprav se prerazporeditev zrn in lezenje pričenja dogajati že v fazi primarne konsolidacije, le da je zaradi stiskanja vzorca manj opazno.

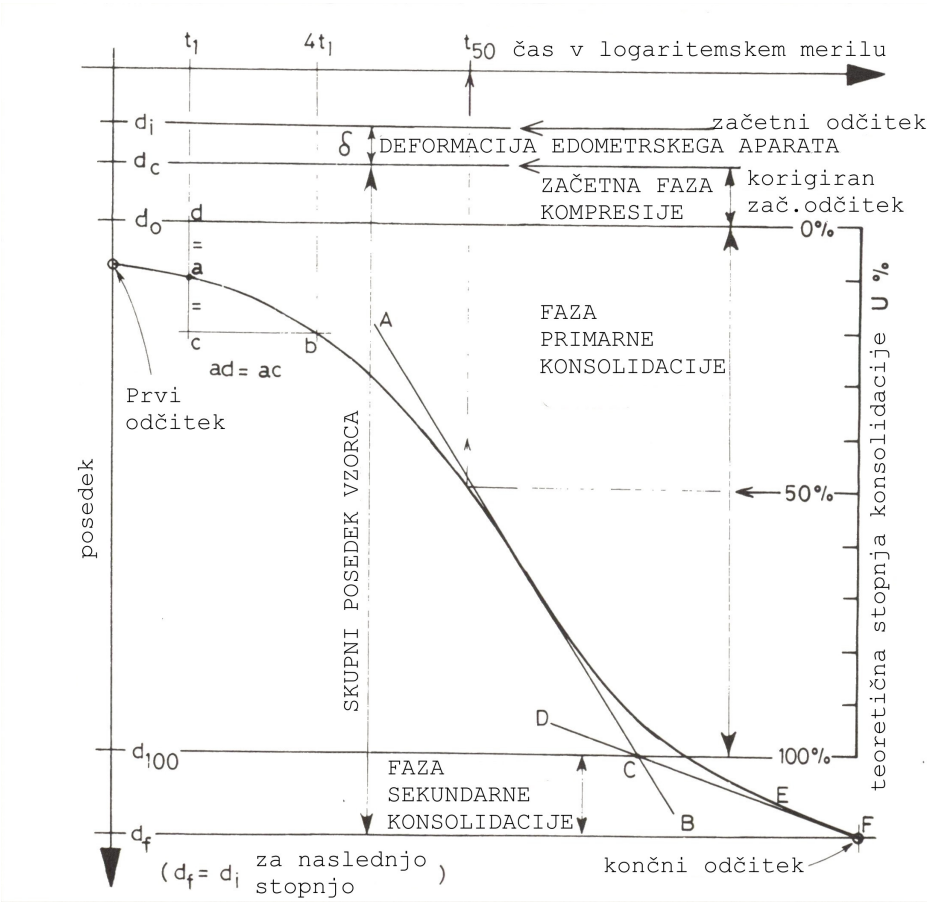


Slika 14: Prikaz uporabe Taylorjevega postopka v programu Edometer V2

Taylorjev postopek za določitev konca primarne konsolidacije je v programu *Edometer V2* izveden tako, da uporabnik na krivulji s klikom miške najprej določi linearni interval krivulje. Na sliki 14 ga omejujeta rdeči točki na krivulji. Program izračuna regresijsko premico med točkami linearnega dela. Nato k dani regresijski premici po zgoraj navedenem postopku določi premico, ki seka krivuljo v točki, kjer je stopnja konsolidacije 0.9. Tedaj je vrednost časovnega faktorja 0.848. Ko je stopnja konsolidacije enaka 1.0, je vrednost časovnega faktorja 2.0. Dobljeni čas presečišča torej množimo s faktorjem 2/0.848=2.36 in dobimo čas konca primarne konsolidacije. Program rezultate Taylorjevega postopka izpiše v izhodno datoteko. Zraven izračuna tudi vrednost koeficienta konsolidacije.

Casagrandejev postopek za izračun konca primarne konsolidacije

Za izračun konca primarne konsolidacije se v praksi uporablja tudi metoda, ki jo je vpeljal Casagrande (Casagrande, 1932).

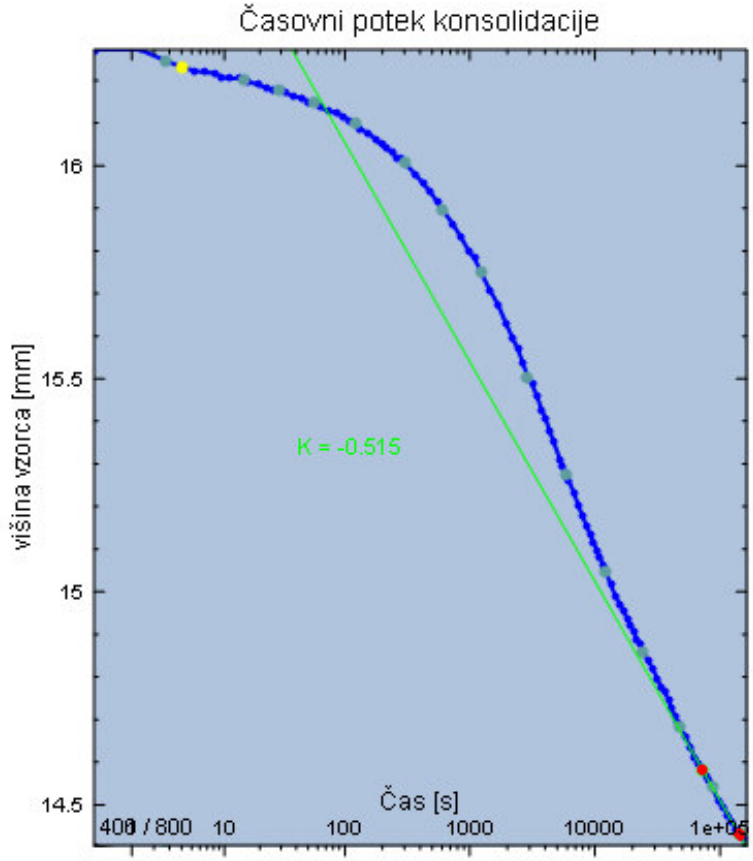


Slika 15: Prikaz Casagrandejevega postopka

Casagrandejeva metoda ima v primerjavi s Taylorjevo slabost, da moramo za ugotovitev konca primarne konsolidacije poznati premico EF (slika 15), ki jo lahko določimo šele, ko se že nahajamo v fazi sekundarne konsolidacije. Pri Taylorjevem postopku pa lahko konec primarne konsolidacije napovemo že pred njenim dejanskim koncem.

Določitev koeficienta sekundarne konsolidacije

Koeficient sekundarne konsolidacije je določen z naklonom tangente na linearni del poteka sekundarne konsolidacije s časom v logaritemskem merilu. Na sliki 15 je koeficient sekundarne konsolidacije naklon premice, ki jo določa premica EF.



Slika 16: Prikaz določitve koeficienta sekundarne konsolidacije v programu

Postopek določitve koeficienta sekundarne konsolidacije prikazuje slika 16. Postopek se izvede tako, da s kazalcem miške določimo mejni točki linearnega dela sekundarne konsolidacije na konsolidacijski krivulji. Program nato izračuna linearno regresijsko premico vseh vmesnih točk, vključno z mejnima. Koeficient regresijske premice je koeficient sekundarne konsolidacije. Vhodni podatki in rezultati postopka se napišejo v datoteko rezultatov.

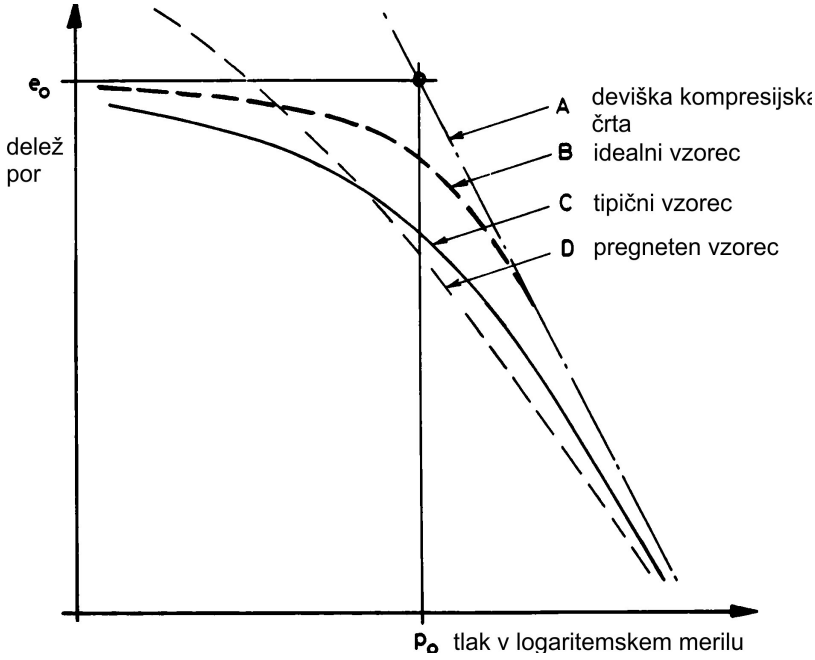
Določitev količnika prekonsolidacije OCR (Overconsolidation ratio)

Ko govorimo o lastnostih konsolidacije zemljin v naravi, lahko zemljine razdelimo v dve skupini:

- normalno konsolidirane zemljine,
- prekonsolidirane zemljine.

Normalno konsolidirane zemljine v geološki preteklosti niso bile izpostavljene večjim pritiskom, kot je trenutni pritisk zaradi lastne teže. V naravi se takšne zemljine pojavljajo kot mladi rečni, jezerski ali morski sedimenti.

Normalno konsolidirane zemljine so zelo občutljive na zunanje motnje, ki lahko bistveno vplivajo na razmerje deleža por v odvisnosti od napetosti. Drugače rečeno, ti materiali zahtevajo pri razcevljanju vzorca iz vrtalne garniture in vgradnji v edometrsko celico posebno pazljivost.



Slika 17: Prikaz vpliva motnje na potek e(σ) krivulje

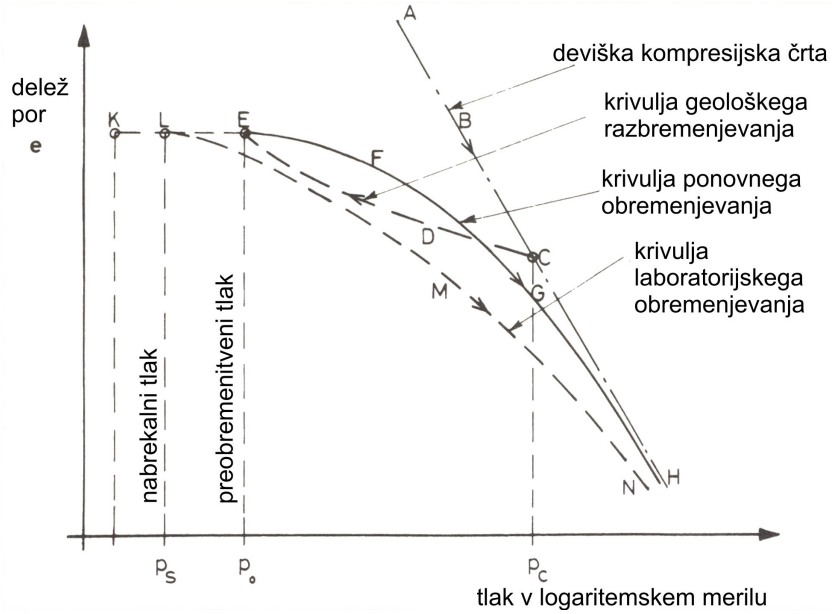
Prekonsolidirane zemljine so bile v geološki preteklosti izpostavljene večjim pritiskom, kot je trenutni geološki tlak. Takšne zemljine se v naravi pogosto pojavljajo na področjih, ki so jih v preteklosti prekrivali ledeniki.

Prekonsolidirane zemljine so za razliko od normalno konsolidiranih manj občutljive na zunanje motnje. Najbolj so občutljive na kontakt z vodo, pri čemer se zmehčajo in nabrekajo. Če nabrekanje konstrukcijsko preprečimo, je lahko konstrukcija izpostavljena zelo velikim nabrekalnim pritiskom. Ko vzorec razcevimo, se zaradi razbremenitve prične relaksirati, kar vpliva na kasnejši potek razmerja količnika por v odvisnosti od napetosti v laboratoriju.

Količnik prekonsolidacije je določen z naslednjo enačbo

$$OCR = \frac{p_c}{p_0} \qquad \text{Enačba 15}$$

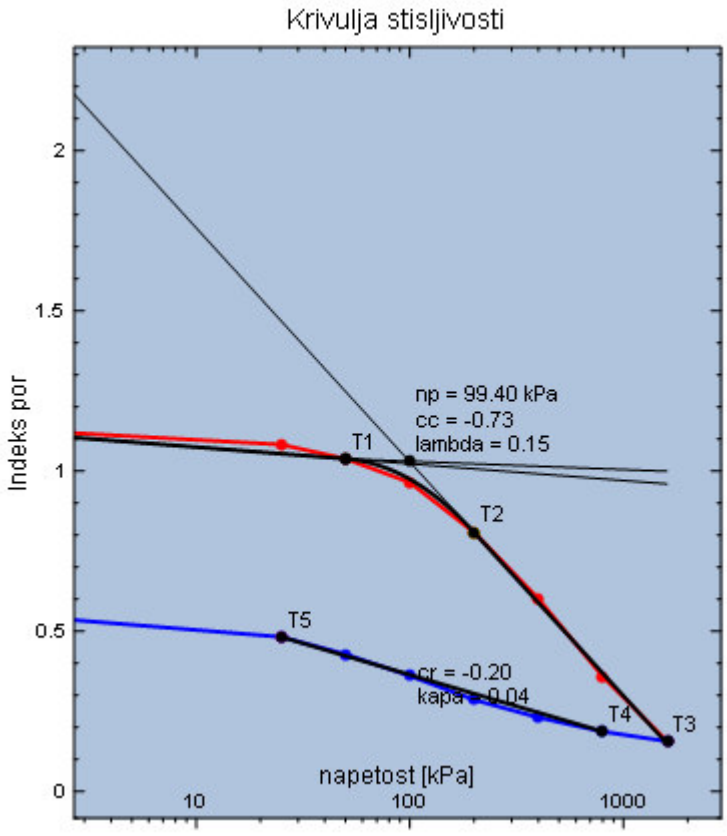
Določitev prekonsolidacijske napetosti je prikazal Casagrande (Casagrande, 1936). Postopek temelji na dejstvu, da je zemljina do prekonsolidacijske napetosti bistveno manj stisljiva, kot je potem. To prikazuje naslednji diagram.



Slika 18: Prikazuje potek e(σ) krivulje razbremenitve in ponovne obremenitve

Slika 18 prikazuje mehanizem določanja prekonsolidacijske napetosti. Linija ABCH prikazuje deviško kompresijsko črto. V preteklosti je bil dosežen tlak p_c. Sledila je razbremenitev na tlak p_0. Če zemljino ponovno obremenimo (krivulja EFG), vidimo, da je krivulja obremenjevanja mnogo položnejša od deviške črte. Ko dosežemo tlak p_c, pa se hitro prične približevati deviški črti.

Napetost p_c lahko s primernim grafičnim postopkom enostavno določimo, kar omogoča tudi program *Edometer V2*.



Slika 19: Postopek določanja OCR v programu Edometer V2

Postopek je izveden po Casagrandejevi metodi določanja OCR. Najprej izberemo točke T1, T2 in T3. Program izračuna regresijsko premico vseh točk levo od T1 vključno s T1. Izračuna tudi regresijsko premico med točkama T2 in T3, vključno z njima. Med premicama iz točk T1 in T2 potegne kubično parabolo. Na mestu največje ukrivljenosti parabole izračuna tangento in kot med njo in horizontalo. Nato pod polovičnim kotom iz točke največje ukrivljenosti potegne premico in najde presečišče med premico T2T3. Vrednost abscise presečišča je iskana prekonsolidacijska napetost.

Matematično dejstvo je, kar je potrdila tudi praksa, da je kubična parabola najbolj ukrivljena bodisi v točki T1 ali T2, odvisno od izbora robnih točk. Kubična parabola torej ni najbolj primerna krivulja za tovrsten postopek.

V praksi se je tudi izkazalo, da je končen izračun zelo odvisen od izbora robnih točk in postavilo pod vprašaj iskanje zapletenejših oblik krivulje z namenom, da bo največja ukrivljenost vedno med točkama T1 in T2.

Postopek iskanja prekonsolidacijske napetosti je delikaten, zahteva veliko izkušenj in v prvi vrsti veliko pazljivost pri ravnanju z vzorcem, saj ima slednje na potek krivulje velik vpliv.

Pri določanju OCR se izračunajo, prikažejo in zapišejo v izhodno datoteko še koeficienti stisljivosti c_c, koeficient razbremenitve c_r, ter koeficienta λ in κ. Za izračun c_r in κ moramo podati še točki T4 in T5. Koeficienta c_c in c_r predstavljata naklona krivulje stisljivosti, kjer napetost nastopa v logaritemskem merilu. c_c je določen kot naklon obremenilne premice T2T3, koeficient cr je določen kot naklon razbremenilne premice T4T5. Pri izračunu koeficientov λ in κ je predpostavljeno troosno napetostno stanje.

$$\lambda = \left| \frac{c_c}{2.3} \right| \qquad \text{Enačba 16}$$

$$\kappa = \left| \frac{c_r}{2.3} \right| \qquad \text{Enačba 17}$$

V zgornjih enačbah c_c predstavlja koeficient stisljivosti, c_r koeficient razbremenitve in e_0 začetni indeks por.

3 ZAHTEVE, IDEJNI NAČRT IN ŠUDIJA IZVEDLJIVOSTI AVTOMATIZIRANEGA MERILNEGA SISTEMA

3.1 Potek edometrske preiskave v KMTAL pred avtomatiziranim merilnim sistemom

Edometrska meritev v KMTAL je pred avtomatiziranim merilnim sistemom potekala s pomočjo elektronskega odčitovalnika konsolidacije vzorca in tudi ročno.

Na elektronski odčitovalnik so priključeni elektronski merilni elementi, katerim se na podlagi merjene vrednosti spremeni upornost, impedanca,... kar odčitovalnik zazna kot spremembo napetosti na elementu. Merilni elementi lahko merijo pomike, tlake, napetosti, temperaturo, deformacijo, itd. V laboratoriju KMTAL se za ta namen uporabljajo odčitovalniki Memo proizvajalca DMS in merilniki pomikov proizvajalca ELE.

Ročni način merjenja je potekal povsem klasično. Laborant je s pomočjo natančne merilne urice s štoparico v roki ob predpisanih časih izmeril pomik na edometrskem aparatu in merjeno vrednost vpisal v formular.

Meritev s pomočjo odčitovalnika mu je v veliki meri olajšala delo. Na začetku nove bremenske stopnje je na merilniku nastavil najkrajši možen vzorčni čas. Ob vklopu odčitovalnika je spustil utež na obremenilni mehanizem in bremenska stopnja se je začela. Ko je preteklo nekaj časa, je na odčitovalniku povečal vzorčni čas in merjenje se je nadaljevalo. Povečevanje vzorčnih časov se je tekom trajanja bremenske stopnje še nekajkrat ponovilo. Čase nastopa bremenske stopnje in merjene vrednosti ob koncu in začetku je laborant za kasnejšo obdelavo in kontrolo ročno beležil v formular.

Ko je bila preiskava končana, je prenesel podatke z notranjega pomnilnika odčitovalnika na računalnik. S pomočjo programa *EdometerFilter* so se podatki najprej razdelili po bremenskih stopnjah in nato filtrirali na čase merjenja, ki jih predpisuje standard. Filtrirani podatki so se nato vnesli v program *Edom2a* (avtor Sebastjan Kuder, univ.dipl.ing.grad.), ki je merjene podatke v končni obliki izrisal na papir.

3.2 Opis merilnega sistema DMS Memo

Merilni sistem *DMS Memo* sestoji iz centralne krmilne enote, na katero so priklopljeni elektronski odčitovalniki. Računalnik komunicira s centralno krmilno enoto, ki komunikacijo posreduje na ustrezen odčitovalnik. Komunikacija med centralno krmilno enoto in računalnikom poteka po standardu RS232 preko običajnih serijskih vrat. Krmilna enota

zagotavlja odčitovalniku poleg komunikacije tudi napajanje. Na krmilno enoto je lahko priklopljenih 128 odčitovalnikov. Omogoča komunikacijo s poljubnim odčitovalnikom, saj ima vsak odčitovalnik svojo sistemsko oznako.

Vsak odčitovalnik ima 3 merilne kanale. Lahko deluje na dva načina; priklopljen na napajanje, temu rečemo *On-Line* način in brez zunanjega napajanja, na *Off-Line* način.

V *On-Line* načinu lahko preko krmilne enote s pomočjo enostavnega krmilnega programa vidimo trenutno vrednost meritve na posameznem kanalu odčitovalnika. Nastavljamo lahko delovne parametre odčitovalnika in beremo ter brišemo notranji pomnilnik. Krmilni program je razvil proizvajalec opreme.

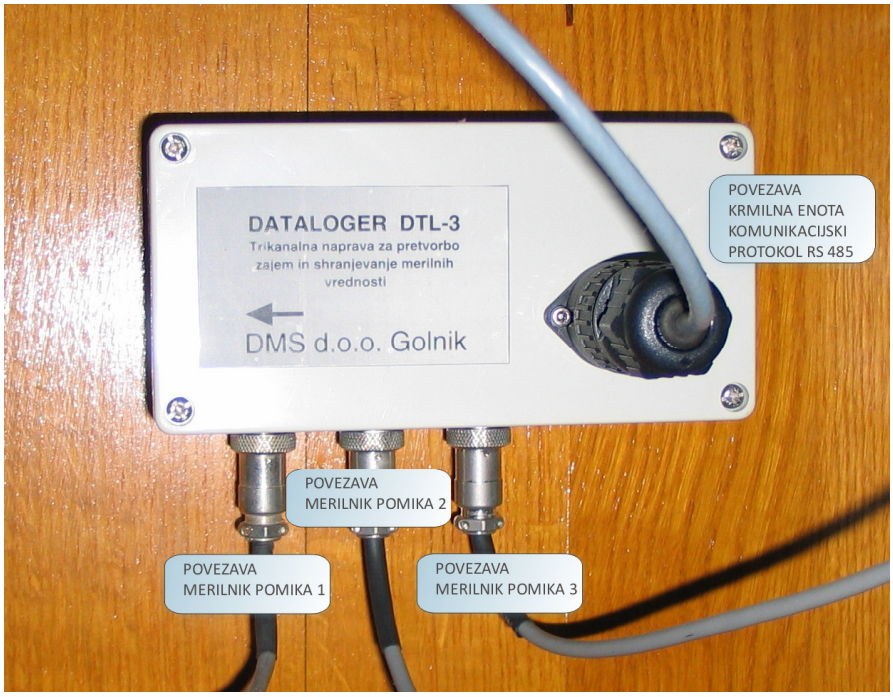
Ko je odčitovalnik v *Off-Line* načinu, se napaja z vgrajenimi akumulatorji in pri nastavljeni frekvenci vzorčenja meri in shranjuje merjene vrednosti vseh kanalov hkrati. Komunikacija s krmilno enoto in z računalnikom takrat ni mogoča. Akumulatorji odčitovalnika se polnijo le, ko se slednji nahaja v *On-Line načinu*. Preklop med *On-Line* in *Off-Line* načinom je izveden ročno preko posebnega stikala, ki se nahaja pri centralni krmilni enoti. To je pravzaprav največja pomanjkljivost sistema.

Analogno digitalni pretvornik (AD) odčitovalnika je 16-biten, kar pomeni, da je njegova resolucija 1/65536. Ima vgrajen tudi IIR (Infinite Impulse Response) filter za filtranje motenj iz okolice, npr. elektromagnetni vpliv omrežja, ki inducira šum z nosilno frekvenco 50 Hz. Ključni vpliv na napako merjenja ima merilnik pomika.

Merilni sistem *DMS Memo* v laboratoriju KMTAL sestoji iz krmilne omare, dveh odčitovalnikov, šestih merilnikov pomika, povezovalnih kablov in krmilnega računalnika. Zaradi prenapetostne zaščite je bil dodan še aktivni sistem brezprekinitvenega napajanja (UPS).

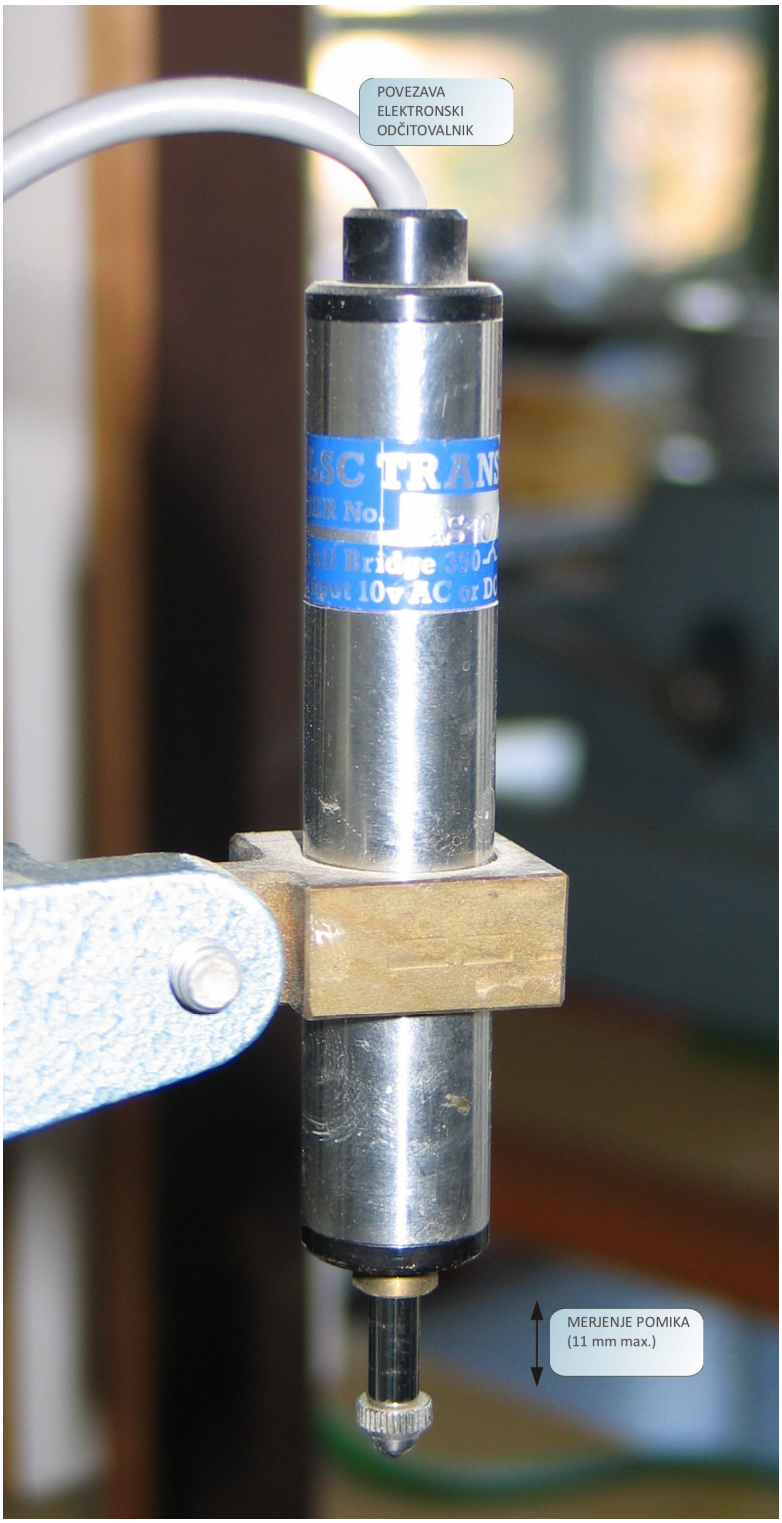
Slika 20: Fotografija krmilne omare DMS Memo v laboratoriju KMTAL

Slika 20 prikazuje krmilno omaro merilnega sistema. V omari se nahajajo glavno stikalo, napajalni sklop, zaščitni sklop, centralna krmilna enota in stikala za *On-Line/Off-Line* preklop odčitovalnikov.



Slika 21: Fotografija elektronskega odčitovalnika DMS Memo

Slika 21 prikazuje elektronski odčitovalnik, ki se nahaja na spodnji strani pulta, na katerega so pritrjeni edometrski aparati.



Slika 22: Fotografija merilnika pomika

Slika 22 prikazuje merilnik pomika vpet na edometrski aparat. Hod merilnega elementa merilnika znaša 11 milimetrov.

3.3 Vrednotenje poteka meritve

Merjenje s pomočjo odčitovalnika je laboranta precej razbremenilo, povečala se je produktivnost, saj je ob ustrezni organizaciji meritev laboratoriju precej povečala zmogljivost pri izvajanju edometrskih preiskav.

Primerjava ročnega načina merjenja z elektronskim pokaže, da je bistvena vendar praktično tudi edina pridobitev elektronskega načina v tem, da laborantu ni treba meriti s štoparico in svinčnikom v roki. Celoten postopek je še vedno odvisen od človeka in s tem posredno izpostavljen različnim napakam. Vsak stroj je prav tako podvržen različnim motnjam in tudi elektronski odčitovalniki niso izjema. Nekajkrat se je zgodilo, da odčitovalnik ni začel z odčitavanjem in je bila več mesecev trajajoča preiskava, izgubljena.

Odčitovalniki *DMS Memo* imajo slabost, da vrednosti v notranjem pomnilniku ni mogoče opazovati v realnem času, ampak jih je treba prenesti v računalnik, jih obdelati in šele potem lahko vidimo, kaj smo izmerili. Ta postopek traja pribl. 10 minut, kar je v praksi precej neugodno.

Tako lahko zaključimo, da je največja slabost pri merjenju z elektronskim odčitovalnikom nezmožnost predstavitve poteka meritve v realnem času. V tem pogledu je ročno merjenje v prednosti, saj lahko časovni potek konsolidacije rišemo sproti. Pri sprotnem risanju časovnega poteka konsolidacije lahko vidimo, kdaj je bremenska stopnja končana in tako prihranimo nekaj časa.

Pri današnji stopnji računalniškega razvoja in stopnji razvoja merilne tehnike so želje po izboljšavi in avtomatizaciji procesa edometrske meritve upravičene. Prva razpoložljiva možnost je nakup sodobnejšega merilnega sistema, ki bi v večji meri zadovoljil potrebe. Žal je ponudba zelo omejena in tesno povezana s strojno opremo in velikimi nabavnimi stroški. Obstoječa strojna oprema še ni amortizirana in njen odpis ne bi bil upravičen. Razvoj lastne programske opreme je tako bolj ekonomičen in tudi strateško zanimiv cilj. Na začetku razvoja je bilo potrebno definirati zahteve merilnega sistema, kar podrobneje prikazuje naslednje poglavje.

3.4 Analiza zahtev avtomatiziranega merilnega sistema

Ključna zahteva pri avtomatizirani edometrski preiskavi je program, ki v realnem času v obliki tabel in grafov (opisanih v prejšnjem poglavju) prikazuje stanje meritve. Izračuna in prikaže naj se vse, kar se da izračunati in narisati. Ker je računska zahtevnost edometrske

preiskave zelo majhna, je zahteva po ponovnem izračunu vsega ob novi izmerjeni točki upravičena in zaradi relativno majhnega števila točk izvedljiva tudi na zelo šibkih računalniških sistemih. Ostale pomembne zahteve so naslednje:

- Urejenost podatkov in rezultatov. Sistem mora biti zasnovan tako, da je rezultate enostavno uporabiti pri reševenju drugih problemov.
- Varnost izmerjenih podatkov v slučaju kolapsa. Edometrska preiskava je dolgotrajna in izguba dela ali vseh podatkov lahko predstavlja poslovno katastrofo.
- Robustnost programa v primeru napak merilnega sistema oz. napak laboranta. Program mora vse strojne napake registrirati in preko alarmov posredovati uporabniku.

3.5 Študija izvedljivosti

Ključne lastnosti sistema *DMS Memo*:

- *On-Line/Off-Line* način delovanja z ročnim preklopom.
- Merjenje vseh 3 kanalov hkrati pri obeh načinih delovanja, (pri *On-Line* načinu lahko izmerimo vrednost vsakega kanala posebej), vendar je praksa pokazala večjo napako merjenih vrednosti v primerjavi z zaporednim merjenjem vseh kanalov.
- Vsi merilni kanali uporabljajo isto uro.
- V *Off-Line* vzorčenju je minimalni interval vzorčenja 15s.
- V *On-Line* načinu traja meritev posameznega kanala slabo sekundo (dejansko se izmerijo vsi trije kanali, posreduje pa se le želena vrednost).
- Pomnilnik odčitovalnika je možno prebrati in pobrisati.
- Možno je nastavljanje in branje trenutnega časa odčitovalnika.
- Branje vsakega merilnega kanala na zahtevo.

Zahteve za strojno opremo za avtomatizirano edometrsko aplikacijo so naslednje:

- Branje vsakega merilnega kanala na zahtevo.
- Standard ob začetku merilne stopnje predpisuje frekvenco merjenja 1 meritev/sek.

Navedene lastnosti merilnega sistema *DMS Memo* v celoti ustrezajo zahtevam avtomatizirane edometrske preiskave.

Komunikacijski protokol med računalnikom in centralno krmilno enoto potreben za izdelavo lastne programske opreme je proizvajalec posredoval. Izdelava lastne edometrske aplikacije je tako postala realen in izvedljiv cilj.

3.6 Izdelava prototipske aplikacije MHTEdometer

Razvoj edometrske aplikacije ni šel po ustaljeni poti razvoja programske opreme, ki praviloma zahteva predhodno teoretično znanje in poznavanje uporabljene tehnologije. Tedaj je treba le določiti primeren model, posamezne elemente načrtati in izdelati sistem.

Pri razvoju zmogljive edometrske aplikacije je bilo najprej potrebno narediti prototip, s pomočjo katerega bi postopoma prišli do potrebnega znanja. Prototipska aplikacija *MHTEdometer* je rezultat dela mojega obveznega praktičnega usposabljanja.

Najprej je bilo potrebno rešiti problem komunikacije s krmilno enoto preko serijskih vrat, kar je zahtevalo približno 50% vsega razvojnega časa. Nadaljnjih 30% časa je bilo porabljeno za razvoj aplikacije in 20% za odpravljanje napak povezanih z neznanjem izdelave tovrstnih računalniško podprtih aplikacij.

MHTEdometer je izvajal meritev in prikazoval časovni potek konsolidacije in diagram poteka edometrskih modulov v realnem času. Ko so bile odpravljene vse napake, je tudi deloval zelo zanesljivo.

Program je izpolnjeval vse ključne razvojne zahteve navedene v prejšnjem poglavju. V uporabi je bil slabi 2 leti, tako da so se lahko odkrile vse pomanjkljivosti in porodile nove posodobitvene ideje.

3.7 Analiza možnih izboljšav s pomočjo prototipa MHT Edometer

Večkrat se je v praksi zgodilo, da je bilo hkrati naročenih veliko število edometrskih preiskav. Razpoložljivih avtomatskih edometrov ni bilo dovolj za vse presikave., zato je bilo potrebno uporabiti tudi ročne merilne aparate. Splošna edometrska aplikacija naj bi tako nudila podporo tudi ročno izvajani preiskavi in z lastno štoparico in zvočnimi signali laboranta opomnila na izvedbo naslednje meritve.

V praksi se je tudi izkazalo, da lahko določene edometrske preiskave trajajo zelo dolgo. Na začetku bremenskih stopenj je potrebno meriti na kratke časovne intervale, ko pa bremenska stopnja traja 1 mesec, se lahko izvede po ena meritev na dan. Ene meritve na dan v nobenem primeru ni težko izvesti ročno, zato je takrat uporaba elektronskega odčitavanja nesmotrna.

Splošna edometrska aplikacija naj bi tako tudi omogočala menjavanje načina merjenja med samo preiskavo.

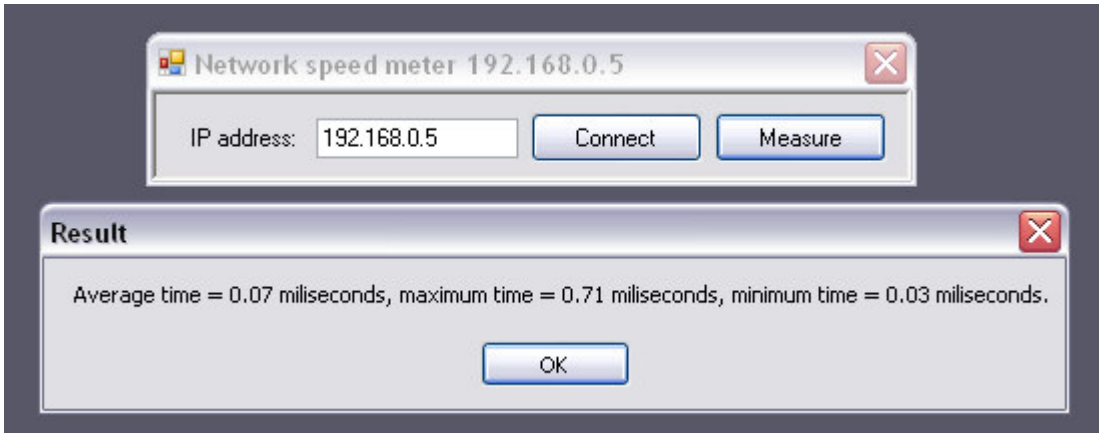
Večkrat se je tudi pojavil problem, da je bilo potrebno poleg edometrskih preiskav opraviti tudi translatorne strižne preiskave, ki so takrat uporabljale iste odčitovalnike. Ko je bila v teku samo ena edometrska preiskava, bi lahko preostala dva merilna kanala porabili za izvajanje merjenja striga, kar takrat ni bilo izvedljivo.

V vsakem primeru potrebujemo tudi za izvajanje sočasne strižne preiskave specializirano aplikacijo. Zaradi same narave delovanja operacijskega sistema računalnika in deljenja virov (serijska vrata) bi tako morali edometrska in strižna preiskava teči znotraj istega procesa, saj uporabljata isti vir. Deljenje serijskih vrat in tudi drugih virov med procesi ni mogoče.

Sklopitev edometrske in strižne preiskave pod isto streho lahko predstavlja, če drugega ne, lokacijski problem v laboratoriju. Zelo neugodno je imeti krmilno aplikacijo in samo fizično izvajanje preiskave v dveh ločenih prostorih. Problem tako lahko rešimo s pomočjo specialnega posebnega programa, ki omogoča deljenje (IPC) izmerjenih podatkov med različnimi procesi (programi).

Današnja razširjenost interneta, cenenost in zmogljivost krajevnih (brezžičnih) omrežij omogoča hiter prenos podatkov in izvajanje ukazov na daljavo. Ukaz na daljavo se izvede sinhrono po naslednjem postopku: na oddaljeno mesto po protokolu TCP/IP pošljemo ukazno sporočilo in počakamo na odgovor.

Če želimo npr. izmeriti vrednost na merilniku pomika pošljemo krmilnemu računalniku kratek ukaz za izvedbo meritve in čakamo na odgovor z rezultatom. Krmilni računalnik tekoči ukaz prepozna, sproži merjenje na merilniku in posredovano vrednost v obliki kratkega sporočila pošlje nazaj.



Slika 23: Prikaz hitrosti ukazov na daljavo pri 100Mbit/s lokalnem omrežju

Slika 23 prikazuje rezultat merjenja hitrosti delovanja povprečja 1000 ukazov na daljavo pri krajevni mrežni povezavi s hitrostjo 100Mbit/s z enostavnim testnim programom. Ukaz je kratko sporočilo, ki ga oddaljeni računalnik le posreduje nazaj.

Test nazorno prikaže, da je z današnjo ceneno tehnologijo možno izvesti sistem, ki deluje na dveh oddaljenih računalnikih zelo odzivno. Edometrska aplikacija, ki bi merjene vrednosti črpala iz oddaljenega računalnika, bi lahko učinkovito rešila vse probleme povezane z lokacijo računalnika, ki upravlja preiskavo in lokacijo računalnika, ki izvaja preiskavo. Tak princip v osnovi zahteva program z več procesi, ki je bila tudi sicer zahtevan pri deljenju istega odčitovalnika med edometrsko in strižno preiskavo.

Merilni sistem *DMS Memo* je sicer soliden dosežek proizvodnje domače merilne tehnike, vendar ima tako kot vsi produkti močno tujo konkurenco, ki npr. nudi merilnike z 24-bitno resolucijo, frekvenco vzorčenja preko 100 meritev na sekundo in z velikim številom merilnih kanalov v kompaktnih izvedbah.

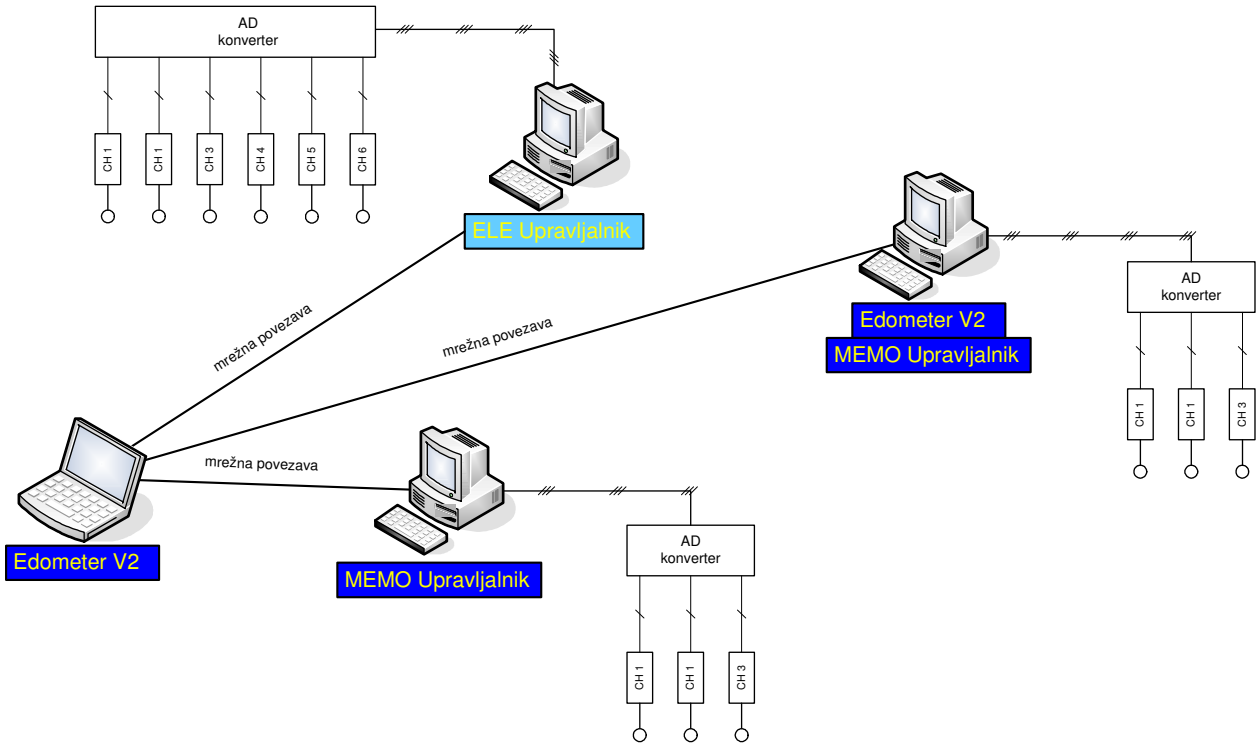
Od merilnih sistemov neodvisne aplikacije so perspektivnejše, saj ostajajo uporabne tudi s prihodom novih tehnologij in jih je primerno sproti razvijati in izboljševati. Za prilagoditev edometrske aplikacije k novi strojni opremi je potrebno zamenjati le krmilni del aplikacije. Komunikacija med edometrsko aplikacijo in krmilno aplikacijo poteka vedno na enak način.

Ideje, ki so se porodile ob uporabi stare edometrske aplikacije *MHT Edometer*, ki naj bi bile podprte v novi, lahko strnemo v naslednjih točkah:

- Edometrska aplikacija naj vsebuje podporo ročno izvajanim preiskavam.
- Edometrska aplikacija naj omogoča med izvajanjem meritve prehod med ročnim in avtomatskim odčitavanjem, oz. menjavo merilne opreme.

- Krmilni program merilnega sistema in edometrska aplikacija sta dva ločena programa.
- Komunikacija med programoma je možna na daljavo.
- Edometrska aplikacija je neodvisna od merilne opreme.
- Merilni kanali odčitovalnika so kljub fizični sklopitvi po sami naravi s pomočjo krmilnega programa navzven prikazani kot popolnoma neodvisni.
- Izdelava izhodnega poročila je zamuden problem, ki ga je primerneje reševati s specializiranimi programi, ki se v laboratoriju za ta namen uporabljajo tudi pri drugih preiskavah (Microsoft Excel). Zahtevana je strukturna urejenost izhodnih podatkov za čim lažje postprocesiranje rezultatov.

Distribuirano delovanje nove aplikacije nazorno prikazuje naslednja shema.



Slika 24: Prikaz distribuiranega delovanja merilnega sistema edometrske aplikacije

Na sliki 24 je prikazana arhitektura distribuiranega delovanja merilnega sistema. Z modro barvo so označeni sklopi sistema, ki so danes izvedeni, z zelenomodro pa sklopi, ki jih je potrebno narediti. Vidimo, da se lahko edometrska in krmilna aplikacija nahajata bodisi na

enem samem računalniku bodisi na dveh. Slika tudi prikazuje, da lahko posamezna edometrska aplikacija uporablja različne merilne sisteme.

4 IZDELAVA PROGRAMOV EDOMETER V2 IN MEMO UPRAVLJALNIK

4.1 Uporabljena orodja

Edometrska aplikacija *Edometer V2* in krmilni program merilne opreme *Memo Upravljalnik* sta bila razvita s pomočjo razvojnega sistema *Microsoft Visual Studio 2005*, v programskem jeziku C# (Liberty, 2005). Manjši del programa *Edometer V2* je bil napisan tudi v jeziku C++. Razvojno okolje *Visual Studio* poleg programskih jezikov C# in C++ podpira tudi jezika Visual Basic.NET in J#. Omogočena je popolna združljivost uporabe komponent napisanih v različnih programskih jezikih.

Obe aplikaciji, *Edometer V2* in *Memo Upravljalnik* sta napisani v objektno usmerjenem načinu programiranja, ki omogoča večjo preglednost in strukturiranost programske kode. V preteklosti so bili programi kodirani v proceduralnem načinu. Proceduralni način pojma objekt ne pozna, deluje na principu globalnih podatkov v programski kodi in procedur (podprogramov), ki s temi podatki upravljajo. Procedure lahko poleg globalnih podatkov uporabljajo tudi argumente (parametre).

Vsi nastavitveni parametri programov *Edometer V2* in *Memo Upravljalnik*, vmesni in končni rezultati se shranjujejo v tekstovnih datotekah po standardu XML.

Za medprocesno komunikacijo na daljavo ali lokalno je bila uporabljena tehnologija *Microsoft Remoting (*Lowy, 2005*)*, ki omogoča učinkovito in enostavno uporabo objektov na daljavo.

Aplikacija *Edometer V2* uporabnika opominja z zvočnimi signali. Zvočno signaliziranje omogoča tehnologija *Microsoft DirectX* oz. njen podsklop *DirectSound. DirectSound* je višjenivojska plast med gonilniki zvočnih naprav in uporabnikom, ki želi narediti zvočno bogato aplikacijo.

4.2 Objektni način programiranja

Objekt oz. predmet v programu je sklop podatkov in operacij nad njimi. Objekt v programu praviloma predstavlja določen predmet v obravnavani nalogi programa. Tudi elementi uporabniškega vmesnika so izvedeni z objekti.

Objekte z enakimi lastnostmi predstavimo s pomočjo razredov. Razred je zapis v programu, ki določa tipe podatkov in operacije objektov istega tipa.

Objekt v programu je predstavljen z objektno spremenljivko določenega tipa (razreda). Vrednost objektne spremenljivke med izvajanjem programa je naslov podatkov objekta v delovnem pomnilniku računalnika (RAM). Objektna spremenljivka je torej kazalec na podatke objekta.

Objektno usmerjeno programiranje (Liberty, 2005) je možno samo s programskimi jeziki, ki ta način podpirajo (C++, C#, Visual Basic .NET, Java, ...).

Objektno programiranje velja za težji način izdelave programov, ker poteka na drugačen način, kot smo sicer vajeni reševati probleme.

Intuitivno rešitev naloge predstavimo kot postopek, zaporedje korakov (operacij), kar ustreza proceduralnemu načinu programiranja. Kompleksnost problemov obvladujemo tako, da proceduro razdelimo na več manjših procedur. Slabost proceduralnega pristopa je v neenakovrednem obravnavanju podatkov in postopkov; kompleksni programi postanejo nepregledni in težko obvladljivi, pri enostavnih programih pa je proceduralni način zelo učinkovit.

Pri objektnem načinu programiranja reševanje naloge modeliramo tako, da identificiramo in razčlenimo objekte, ki v njej nastopajo. Ugotovimo njihove podatke, lastnosti in operacije. Objekte iste vrste razvrstimo v razrede.

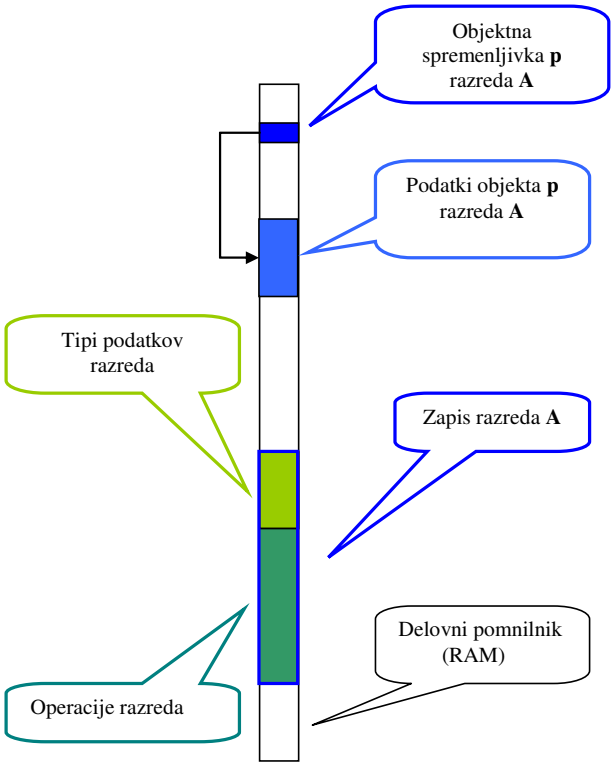
Predmetni model naloge nato s pomočjo objektnega programskega jezika preslikamo v program. Za vsako vrsto objektov v programu je potrebno v program dodati definicijo razreda. Razred je zapis vseh lastnosti predmetov iste vrste.

Razred ima ime, vsebuje definicije podatkov in operacije, ki določajo obnašanje objektov. Pri podatkih je potrebno določiti ime in tip podatka. Operacije zapišemo v obliki funkcij, ki vsebujejo izvedljive stavke. Operacije znotraj razredov običajno imenujemo *metode*. Dostop do podatkov in operacij lahko s posebnimi določili *private*, *protected* in *public* omejimo, kar sistemsko preprečuje njihovo nesmotrno uporabo.

Objekt določenega razreda je v programu fizično predstavljen z zapisom podatkov v glavnem pomnilniku. Vsak objekt ima svoje podatke, ki določajo njegovo tekoče stanje. Operacije nad objekti iste vrste pa so zapisane samo enkrat v definiciji razreda, ki je med izvajanjem programa tudi zapisana v glavnem pomnilniku.

Objekti med izvajanjem programa nastajajo, se nekaj časa uporabljajo in nato uničijo.

Nastanek objekta na določenem mestu programa se v jeziku C# zapiše s posebnim operatorjem *new*, ki ustvari objekt določenega razreda. Točneje rezervira ustrezen prostor za podatke objekta in nato vanje zapiše začetno stanje teh podatkov. Ustvarjeni objekt je potrebno prirediti objektni spremenljivki, ki je tipa z istim imenom kot razred prirejenega objekta. Vrednost objektne spremenljivke je naslov začetka podatkov v glavnem pomnilniku. S pomočjo objektne spremenljivke lahko v kodi programa dostopamo do podatkov objekta in sprožimo operacije nad objektom.



Slika 25: Prikaz primera objekta in razreda v pomnilniku

Za uničenje objektov skrbi pri novejših objektnih programskih jezikih (C#, Visual Basic.NET, java, ...) poseben podsistem, ki se v platformi .NET imenuje *garbage collector*, ki samodejno odstranjuje objekte (iz glavnega pomnilnika), ko se nanje ne sklicuje nobena objektna spremenljivka. Povezavo med objektno spremenljivko in prirejenim predmetom prekinemo na enostaven način, tako da ji priredimo ničelni naslov, vrednost *null* (C#).

4.2.1 Primer razreda v programu Edometer V2

Pri edometrski preiskavi je *bremenska stopnja* velikokrat uporabljena beseda. Ob začetku nove bremenske stopnje določa masa uteži na obremenilnem mehanizmu navpično napetost, ki ji je vzorec izpostavljen. Vse meritve stiskanja vzorca so izvršene v dani bremenski stopnji

oz. pri dani napetosti. Za vsako bremensko stopnjo tudi sleherni trenutek vemo, koliko merjenj stiskanja vzorca je bilo izvedenih. Običajno ima vsaka edometrska meritev več bremenskih stopenj in vse imajo enake podatke in operacije.

Pojem bremenske stopnje je iz prakse zelo enostavno preslikati v pojem razreda v programu.

Razred z imenom *BremenskaStopnja* mora torej imeti naslednje podatke:

- napetost, pri kateri se merjenje izvaja,
- seznam izmerjenenih točk.

Razred z imenom *BremenskaStopnja* mora imeti naslednje operacije nad podatki:

- DodajTočko - doda novo merjeno točko v seznam merjenih vrednosti.
- OdstraniTočko - odstrani izbrano merjeno točko iz seznama.
- VrniTočko - vrne izbrano merjeno točko seznama.
- ŠteviloTočk - prešteje in vrne število izmerjenih vrednosti v seznamu.

Pri objektnem načinu programiranja najprej v programskem jeziku določimo definicijo razreda *BremenskaStopnja*, ki vsebuje dva dela: podatke in operacije. Podatkovni del določa stanje objekta (bremenske stopnje), operacije pa so postopki, ki to stanje spreminjajo. Ko imamo zapisano definicijo razreda, lahko ustvarimo enega ali več objektov - primerkov ali instanc razreda *BremenskaStopnja*, katerim potem s pomočjo operacij (metod) dodajamo oz. odvzemamo izmerjene točke, jih preštejemo, pregledamo, itd. Uporabo objektov v programski kodi lahko predstavimo na naslednji način:

Nekje v kodi ustvarimo objekt z imenom *bs* razreda *BremenskaStopnja*. Točneje najprej definiramo spremenljivko *bs* tipa *BremenskaStopnja*, nato z operatorjem *new* ustvarimo objekt razreda *BremenskaStopnja* in ga priredimo spremenljivki. Če želimo potem v objekt *bs* dodati novo merjeno točko, napišemo: *bs.DodajTočko()*. Dodana točka običajno pride v bremensko stopnjo kot vhodni parameter te operacije: *bs.DodajTočko(točka)*. Če želimo ugotoviti število točk v *bs* napišemo: *bs.ŠteviloTočk()* in število točk dobimo kot rezultat te operacije.

4.3 Podatkovni standard XML

Podatkovni standard *XML (Extensible Markup Language)* določa standardni zapis podatkov praviloma v obliki dokumentov, ki so vedno zapisani v obliki besedila (W3C XML 1.0:2006).

Podatki so strukturirani v drevesasto strukturo elementov XML. Na vrhu je zato vedno en sam element XML imenovan *izhodiščni* ali *dokumentni element* XML. Elementi XML vsebujejo označene podatke.

Element XML je sestavljen iz treh delov: začetne oznake, vsebine in končne oznake. Začetna oznaka je ime elementa XML v zašiljenih oklepajih. Končna oznaka se od začetne razlikuje samo v dodani poševnici, ki sledi začetnemu znaku <. Na primer:

<BremenskaStopnja>vsebina </BremenskaStopnja>.

Vsebina elementa XML sme biti bodisi podatek bodisi niz podelementov XML. Na primer, pri elementu <X>2.5</X> je vsebina podatek 2.5, pri elementu <Točka><X>2.5</X><Y>-3</Y></Točka> pa je vsebina <X>2.5</X><Y>-3</Y>.

Podatki v elementu XML pa smejo biti zapisani tudi v obliki atributov XML, ki so vedno del začetne oznake elementa XML. Atributov sme biti več, njihov vrstni red ni pomemeben. Vsak element XML lahko vsebuje poleg vsebine tudi podatke v obliki atributov.

Atribut XML je vedno sestavljen iz imena, ki mu sledita enačaj in vrednost atributa med enojnimi ali dvojnimi narekovaji. Dovoljeno je, da je element XML bodisi brez atributov bodisi ima prazno vsebino med začetno in končno oznako. Elemente XML z atributi in brez vsebine smemo zapisati v okrajšani obliki brez končne oznake, na primer:

<Točka relativniČas="0" odčitek="-3.558799" />.

Kot vidimo na primeru, je potrebno dodati takemu elementu v začetni oznaki pred zašiljenim zaklepajem poševnico.

Element XML brez vsebine je torej določen z imenom in podatki, ki so zapisani v atributih. Primer dokumenta podatkov zapisanih po standardu XML, ki predstavlja bremensko stopnjo je naslednji:

```
<?xml version="1.0" encoding="windows-1250"?>
<BremenskaStopnja napetost="150" časZačetka="10. 10. 2007 18:24:42.826">
    <ČasovniProtokol meje="3 15 30 60" intervali="0 2 5 10" />
    <Točka relativniČas="0" odčitek="-3.558799" />
    <Točka relativniČas="1.252" odčitek="-3.557551" />
    <Točka relativniČas="2.454" odčitek="-3.455212" />
</BremenskaStopnja>
```

Programiranje procesiranja dokumentov XML (čitanje, pisanje in urejanje) poteka pri vseh programskih jezikih in platformah na enak način. Napredni objektno usmerjeni programski jeziki (C#, Visual Basic.NET, Java, ...) imajo že izdelane programske komponente za obravnavo dokumentov XML. Večja učinkovitost programiranja podatkov je bil eden izmed razvojnih ciljev standarda XML.

Pri programu *Edometer V2* in *Memo Upravljalnik* struktura podatkov v datoteki XML verno sledi objektni strukturi programa.

4.4 Microsoft Remoting

Za medprocesno komunikacijo med merilno aplikacijo in merilnim programom je bila uporabljena razvojna tehnologija *Microsoft Remoting (*Lowy, 2005*). Remoting*, ki omogoča v programih uporabo objektov na daljavo. Določa protokol izmenjave podatkov o objektih. Na primer, na računalniku A imamo objekt *bs* razreda *BremenskaStopnja*. S pomočjo tehnologije *Remoting* je omogočena oddaljena raba objekta *bs*.

Na računalniku B pripravimo spremenljivko razreda *BremenskaStopnja* z imenom *bsOddaljena* in s pomočjo tehnologije *Remoting* spremenljivki *bsOddaljena* določimo enakovredno instanco, kot jo ima objekt *bs*. Stavek *bsOddaljena.DodajTočko* izveden na računalniku *B* doda novo točko v kontejner bremenske stopnje, ki se fizično nahaja na računalniku *A*.

Uporabo tehnologije *Remoting* v objektnem programu lahko predstavimo na naslednjem primeru:

V programu, ki se izvaja na računalniku *B,* želimo izvedeti, koliko točk vsebuje objekt *bsOddaljena*., ki se nahaja v programu izvajanem na računalniku A. Stavek *bsOddaljena.ŠteviloTočk* pošlje računalniku *A* kratko sporočilo in čaka na rezultat. Ko sistem *Remoting* na računalniku A sporočilo sprejme, pokliče *bs.ŠteviloTočk* in dobljen rezultat kot odgovor pošlje nazaj računalniku B. Računalnik B dobi to sporočilo in ga upošteva kot rezultat operacije *bsOddaljena.ŠteviloTočk*.

Sistem *Remoting* je torej urejen sistem pošiljanja informacij o objektih na daljavo, s katerimi se razvijalcu programa ni potrebno podrobno ukvarjati. Razvoj objektnega programa z objekti na daljavo poteka na popolnoma enak način, kot pri enovitem samostojnem programu.

Če želimo objekt uporabljati na razdaljo preko tehnologije remoting (ali katerekoli druge), mora biti možno objekt serializirati.

Serializacija je proces zapisa tekočega stanja objekta v strnjen niz (zaporedje) podatkov. Lahko je to bodisi niz binarnih podatkov bodisi zapis podatkov v obliki besedila. Običajno je za tekstoven zapis uporabljen podatkovni standard XML, lahko pa je uporabljen tudi kak drug tekstnovni zapis. Zahtevano je le, da je proces serializacije in deserializacije enoličen.

Deserializacija je obraten postopek od serializacije. Ko sistem *Remoting* pošlje serializiran objekt na oddaljen računalnik, ga je tam potrebno spet pretvoriti v normalen objekt. Seveda mora biti isti razred, kot ustreza poslanemu objektu, definiran tudi na oddaljenem računalniku.

Če objekt razreda A serializiramo na računalniku X in ga nato deserializiramo na računalniku Y, moramo dobiti enakovreden objekt razreda A z enakim stanjem, kot je bilo pri objektu v trenutku serializacije.

Razvojno okolje *Visual Studio* in izvajalno okolje .NET imata za najbolj pogoste tipe podatkov že vgrajene zmogljivosti bodisi za tekstovno serializacijo po standardu XML bodisi za binarno serializacijo. V primeru objektov specialnih podatkovnih tipov je možno za take razrede serializacijo in deserializacijo sprogramirati posebej.

4.5 Microsoft DirectX oz. DirectSound

Tehnologija *Microsoft DirectX* je namenjena izdelavi grafično in zvočno bogatih programov. Sestavlja jo več podsklopov. Za procesiranje zvoka skrbi sklop *DirectSound (*Fay et al.,2003). *DirectX* se sicer največ uporablja pri izdelavi računalniških igric, kjer je danes dosežen zelo soliden nivo navidezne resničnosti. Visoko zmogljivost tehnologije *DirectX* se da s pridom uporabiti tudi v tehničnih aplikacijah.

Ključna korist uporabe tehnologije *DirectX* je v tem, da se uporabniku ni niti potrebno ukvarjati z gonilniki strojne opreme niti ni potrebno poznati ozadja delovanja računalniške grafike oz. procesiranja zvoka.

DirectX hkrati tudi poskrbi, da se posamezne računsko zahtevne grafične oz. zvočne operacije izvajajo ločeno na procesorjih grafičnih oz. zvočnih kartic. Tako zelo razbremeni glavni procesor računalnika, ki lahko več časa posveti reševanju zastavljene naloge. Za uporabo celotne funkcionalnosti tehnologije *DirectX* je potrebno poznavanje programskega jezika C++.

4.6 Problem večopravilnosti

Programi, ki samodejno izvajajo neko nalogo in hkrati omogočajo dodatno uporabniško aktivnost (na primer uporabnika sproti obveščajo o stanju izvajanja), so večopravilni oz. večnitni programi (Lowy, 2005). Takšen je tudi program *Edometer V2,* v katarem hkrati tečeta dve glavni izvajalni niti.

Izvajalna nit je zaporedje ukazov, ki ga izvaja procesor računalnika. V programu je zapisana v obliki procedure, ki vsebuje zaporedje izvedljivih stavkov. Izvajalna nit se sproži ob klicu izvajalni niti prirejene procedure in se konča ob njenem zaključku. V objektnem programu je lahko ta procedura metoda nekega razreda.

Računska enota procesorja računalnika lahko ob določenem času izvaja le eno izvajalno nit, ostale čakajo na izvajanje. Procesiranje izvajalnih niti krmili (preklaplja) operacijski sistem računalnika. Kadar si dve ali več izvajalnih niti delijo iste podatke oz. objekte, je potrebno spreminjanje podatkov oz. objektov sinhronizirati.

Izdelava večnitnih programov je bolj zapletena od enonitnih. Napake, ki so posledica neustrezne sinhronizacije izvajalnih niti, je zelo težko odkriti, ker se včasih zgodijo, včasih ne. Težko je tudi predvideti njihove posledice, ki so od računalnika do računalnika različne. Kar na počasnem računalniku X deluje brezhibno, lahko na hitrem računalniku Y odpove, saj se izvajalne niti lahko vklapljajo v drugačnem (konfliktnem) vrstnem redu. V programerski praksi velja načelo *izogibanja večnitnosti, če se le da*.

Obstaja veliko uporabnih večnitnih programov. Zelo uporabljana programa sta npr. *Raziskovalec* v operacijskem sistemu *Windows* in program *Media Player* za igranje multimedijskih vsebin. Obstaja tudi veliko enonitnih programov kot so npr. AutoCAD, Plaxis, Amses, Okvir, itd.

Prvi niti v programu *Edometer V2* lahko rečemo *merilna nit*. *Merilna nit* na zelo kratke časovne intervale meri trenutni čas in pošlje krmilnemu programu ob primernem času ukaz za izvedbo nove meritve, nato čaka na rezultat. Ko je rezultat dobljen, merjeno vrednost doda v kontejner zadnje bremenske stopnje.

Drugi niti v programu *Edometer V2* lahko rečemo *okenska nit*, ki obravnava premike in klike miške, pritiske na gumb tipkovnice, splošno rečeno, skrbi za uporabniški vmesnik.

Medtem, ko npr. spreminjamo vrednost 10. točke v bremenski stopnji 25/50, se merilna nit nemoteno izvaja in ob ustreznem času doda novo meritev. Uporabnik lahko npr. ob

poljubnem času odstrani 15. meritev zadnje bremenske stopnje. Če bi se odstranjevanje zgodilo ravno takrat, ko merilna nit vpisuje novo vrednost, bi lahko prišlo do napake z nepredvidljivimi posledicami, saj bi bilo število meritev v tistem trenutku napačno.

4.7 Splošna načela izvedbe programov

Praksa pri izdelavi programa *MHTEdometer* je pokazala, da je potrebno za večjo preglednost programske kode ločiti objekte uporabniškega vmesnika od izvedbenih objektov.

Pri zasnovi programa *Edometer V2* je bilo narejenih tudi nekaj manjših sprememb pri objektni strukturi programa. Najpomembnejša je bila ta, da ima vsak objekt v programu tudi referenco (sklicevanje) na svojega lastnika. Lastnik objekta v programu je običajno objekt, ki je bodisi obravnavani objekt ustvaril bodisi ima kontrolo nad njim. S tem je na zelo enostaven način omogočeno obojesmerno plezanje po drevesasti strukturi objektov. Objektna struktura je podrobneje predstavljena v naslednjem poglavju.

4.8 Ključni sistemski problemi aplikacije, ki jih je bilo potrebno rešiti

V poglavju predstavitve obstoječega merilnega sistema je navedena možnost *Off-Line* delovanja odčitovalnika, ki je omogočena v programu *Edometer V2*. Slednja je razvoj programa nekoliko zapletla, saj je vnesla dva ključna problema:

- problem časovne zveznosti in
- problem razdružitve tesno sklopljenih merilnih kanalov v *Off-Line* delovanju.

4.8.1 Problem časovne zveznosti:

Ko poteka meritev v *On-Line* načinu, določa čas meritve ura računalnika. Čas meritve se izračuna kot sredinski čas med časom začetka klica funkcije, ki izmeri vrednost izbranega kanala in časom njenega zaključka. Ko merjenje postavimo v *Off-Line* način, postane časovna baza meritve ura odčitovalnika, ki seveda teče malo drugače kot ura računalnika. Ko preklopimo spet v *On-Line* način, se lahko zgodi, da je ura na odčitovalniku prehitevala in v tem primeru lahko dobimo negativno časovno razliko med izmerjenimi točkami. To se seveda nikdar ne sme zgoditi.

Problem rešimo tako, da si pred preklopom v *Off-line* način zapomnimo časovno razliko med urama meritve v računalniku in odčitovalniku. Po preklopu na *On-line* način uro meritve v računalniku nastavimo na uro odčitovalnika z upoštevanjem časovne razlike.

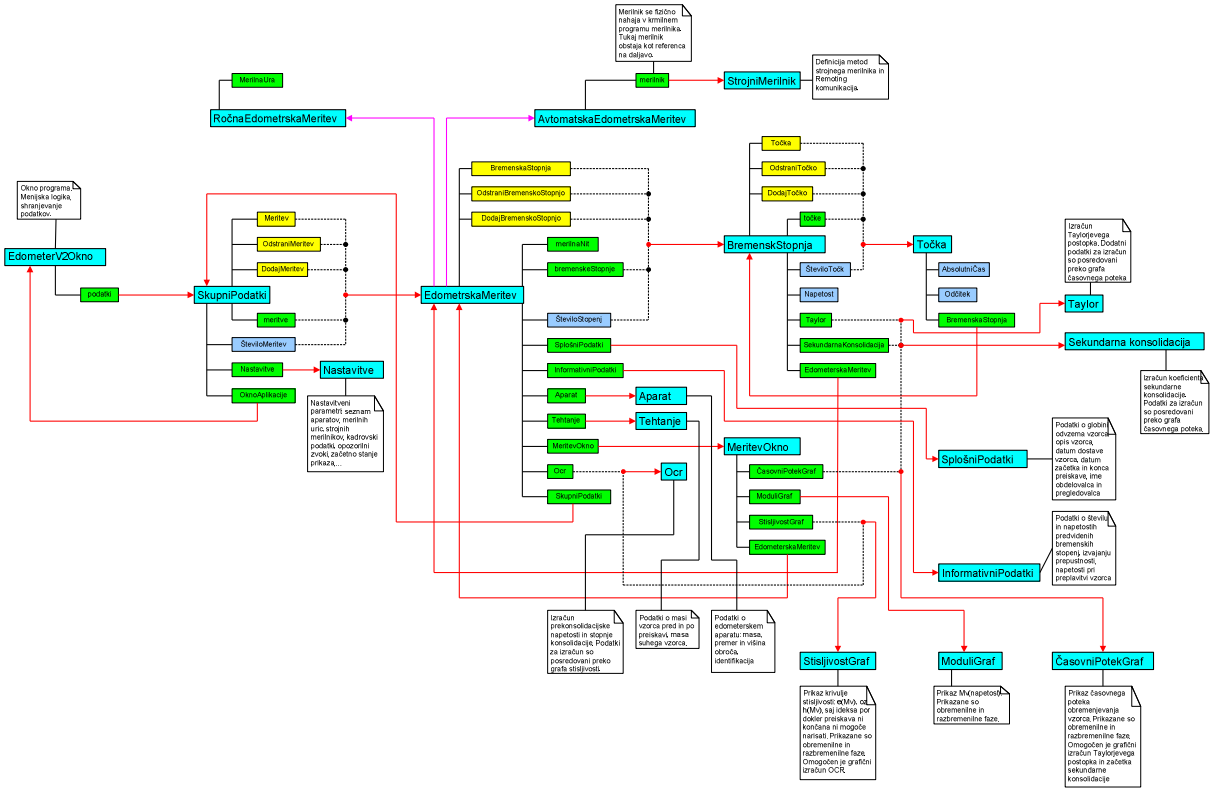
4.8.2 Problem razdružitve sklopljenih merilnih kanalov:

Ko je odčitovalnik v *Off-line* delovanju, so vsi trije kanali merjeni hkrati z isto predpisano frekvenco vzorčenja. Po samem konceptu merilnega sistema, ki zahteva med seboj neodvisne merilne kanale, pa lahko pri eni edometrski meritvi zahtevamo *Off-line* interval vzorčenja 5 min, v drugi pa 30 min. Krmilni program merilnika mora tako *Off-line* vzorčenje odčitovalnika nastaviti na najmanjši skupni deljitelj vseh zahtevanih časov *Off-line* vzorčenja. Ob prehodu na *On-line* način pa je potrebno edometrskim aplikacijam poslati le tiste meritve, ki ustrezajo času *Off-line* vzorčenja.

5 IZVEDBA PROGRAMOV

5.1.1 Edometer V2

Program *Edometer V2* je izveden z upoštevanjem analize problemov in tehnologij iz prejšnjega poglavja.



Slika 26: Grobi objektni model programa Edometer V2

Slika 26 prikazuje grobi objektni model programa *Edometer V2*.

- Modra barva predstavlja definicijo razreda.
- Zelena barva določa objekt oz. spremenljivko, ki ga predstavlja.
- Z rumeno barvo so označene metode oz. funkcije razreda.
- Svetlo modra barva označuje lastnosti razreda.
- Rdeča puščica povezuje objektno spremenljivko z definicijo razreda.
- Vijolična črta označuje izpeljano definicijo razreda.
- Črtkana črta, ki povezuje lastnosti in metode, nakazuje, nad katerim objektom se slednje izvajajo.
- Črtkana črta, ki povezuje objekte, nakazuje njihovo tesno povezanost pri izvajanju nalog.

Ultimativni objekt, ki vsebuje vse objekte programa, določa razred *SkupniPodatki*. Program lahko hkrati izvaja več različnih preiskav. Iz tega sledi, da mora objekt razreda *SkupniPodatki* vsebovati seznam elementov razreda *EdometrskaMeritev*.

Poznamo edometrsko meritev z ročnim in elektronskim odčitavanjem. Obe se razlikujeta le v načinu merjenja, velika večina podatkov je skupnih. Nesmotrno je stvari ponavljati in pisati posebej definicijo razreda *RočnaEdometrskaMeritev* in razreda *AvtomatskaEdometrskaMeritev*. Objektno programiranje omogoča dedovanje razredov. Iz določenega razreda lahko izpeljemo drugega. Javni uporabniški vmesnik izpeljanega razreda podeduje javno dostopne lastnosti in obnašanje (metode) osnovnega razreda. Tako je enostavneje definirati razred *EdometrskaMeritev* in razreda *RočnaEdometrskaMeritev* in *AvtomatskaEdometrskaMeritev* izpeljati iz njega. V izpeljanih razredih potem le še določimo specifične podatke in obnašanje.

Pri vsaki edometrski meritvi v praksi naredimo več bremenskih stopenj. Zato tudi razred *EdometrskaMeritev* vsebuje seznam bremenskih stopenj. Vsaka edometrska meritev se izvaja v svojem edometrskem aparatu. Zato tudi razred *EdometrskaMeritev* vsebuje sklicevanje na objekt razreda *Aparat*. O vsakem vzorcu edometrske meritve moramo vedeti, na kateri lokaciji je bil vzorec odvzet, na kateri globini, kdaj, itd. Razred *EdometrskaMeritev* tako

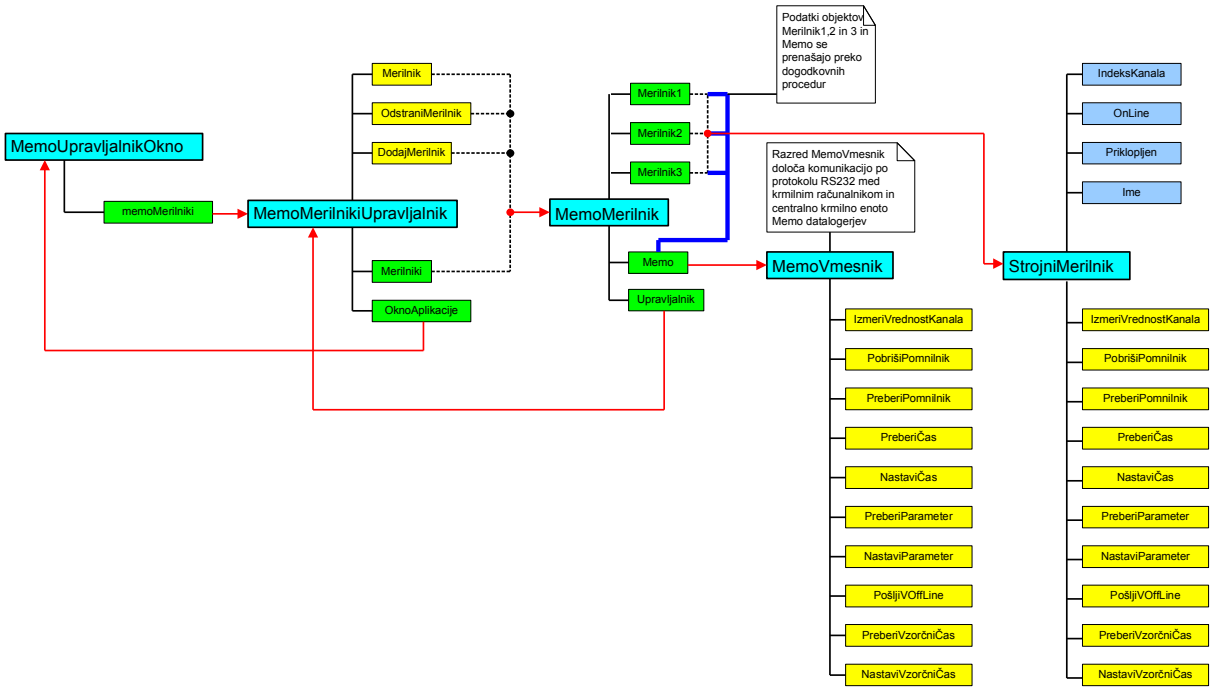
vsebuje tudi sklicevanje na objekt razreda *SplošniPodatki*. Razred EdometrskaMeritev določa tudi mehanizem časovne izravnave, katere problem je opisan v prejšnjem poglavju.

V prejšnjih poglavjih je bil razložen grafični postopek ugotavljanja prekonsolidacijske napetosti. Izračun tega postopka sprožimo ob prikazu krivulje stisljivosti. Da lahko postopek izvedemo, potrebujemo določene podatke vseh bremenskih stopenj. Zato je najpametneje računski objekt postopka, ki ga določa razred *Ocr* (overconsolidation), vgraditi v samo definicijo razreda *EdometrskaMeritev,* saj so tako podatki neposredno na razpolago. Objekt razreda *StisljivostGraf*, ki poskrbi za izris krivulje stisljivosti in izris rezultatov postopka ugotavljanja prekonsolidacijske napetosti, pa se na računski objekt razreda *Ocr* sklicuje, kar v diagramu slike 26 prikazuje črtkana črta.

Objektni model lahko tako razčlenjujemo do vseh najmanjših podrobnosti. Razvidno je, da objektni model sledi operacijam in podatkom, o katerih moramo voditi evidence, če preiskavo izvajamo povsem ročno s svinčnikom in papirjem. Objektno programiranje se zelo izkaže, saj celotno izvajanje programa in sama objektna struktura sledi realnemu poteku edometrske preiskave.

5.1.2 Memo Upravljalnik

Program *Memo Upravljalnik* je po zgradbi nekoliko preprostejši, uporablja manj različnih tehnologij kot *Edometer V2*. Njegov objektni model je bolj abstrakten, ker je tudi naloga, ki jo program opravlja, očem skoraj nevidna.



Slika 27: Grobi objektni model programa Memo Upravljalnik

Slika 27 prikazuje grobi objektni model programa.

- Modra barva predstavlja definicijo razreda.
- Zelena barva določa objekt oz. spremenljivko, ki ga predstavlja.
- Z rumeno barvo so označene metode oz. funkcije razreda.
- Svetlo modra barva označuje lastnosti razreda.
- Rdeča puščica povezuje objektno spremenljivko z definicijo razreda.
- Vijolična črta označuje izpeljano definicijo razreda.
- Črtkana črta, ki povezuje objekte nakazuje njihovo tesno poveznaost pri izvajanju nalog.
- Modra črta nakazuje sklopljenost razredov preko dogodkovnih procedur

Ključni razredi programa so *MemoMerilnik*, *MemoVmesnik* in *StrojniMerilnik*.

Razred *MemoMerilnik* določa objekt *Memo* razreda *MemoVmesnik* za komunikacijo s posameznim odčitovalnikom sistema *Memo*. Poleg teh določa še tri objekte: *Merilnik1*, *Merilnik2* in *Merilnik3* razreda *StrojniMerilnik*. Preko slednjih program *Edometer V2* črpa merjene podatke. V razredu *MemoMerilnik* je določena tudi rešitev problema razdružitve sklopljenih merilnih kanalov v *Off-Line* načinu delovanja prikazanem v prejšnjem poglavju.

V razredu *MemoVmesnik* je definirana vsa komunikacija s centralno krmilno enoto odčitovalnikov. Komunikacija poteka po standardu RS232 preko serijskih vrat. Komunikacija deluje po principu pošiljanja ukaza in čakanja na odgovor. Centralni enoti pošljemo urejen niz bajtov in čakamo rezultat tipa niz bajtov. Centralna enota niz prepozna, izvede ustrezno operacijo in vrne rezultat. Za izvedbo memo vmesnika je potrebno poznati komunikacijski protokol med računalnikom in centralno krmilno enoto sistema *Memo*. *MemoVmesnik* tako določa operacije branja vrednosti posameznega kanala odčitovalnika, branja in brisanja spomina, branja in nastavljanja ure odčitovalnika, itd.

Objekti razreda *StrojniMerilnik* so nosilci podatkov med programom *Memo Upravljalnik* in programom *Edometer V2*, prenos pa omogoča sistem *Remoting*. Razlog obstoja objektov razreda *StrojniMerilnik* je v tem, da lahko preko sistema *Remoting* prenašamo le objekte, ki jih lahko serializiramo. Objektov razreda *MemoVmesnik* se zaradi povezanosti s strojno

opremo ne da serializirati. Objekti razreda *StrojniMerilnik* kličejo preko odzivnikov dogodkovnih procedur v razredu *MemoMerilnik* funkcije objekta *Memo*.

6 OPIS FUNKCIONALNOSTI AVTOMATSKEGA MERILNEGA SISTEMA IN GROBI PRIKAZ UPORABE

6.1 Ključne funkcionalnosti merilnega sistema Edometer V2 in Memo Upravljalnik

Število edometrskih meritev, ki jih program Edometer V2 hkrati izvaja, je teoretično omejeno z velikostjo RAM pomnilnika računalnika. Pri praktični uporabi je preglednost grafičnega uporabniškega vmesnika faktor, ki omejuje število hkratnih meritev.

Edometer V2 je od strojnega merilnega sistema neodvisen. Hkrati lahko uporablja različne vrste merilnih sistemov, saj je krmilni program merilnega sistema tisti, ki podatke na ustrezen način posreduje edometrski aplikaciji.

Podprte so avtomatske, kot tudi ročno vodene preiskave. Podprta je tudi možnost spreminjanja načina merjenja med izvajanjem preiskave za čim bolj ekonomično rabo merilne opreme.

Izvedeno je zvočno opozarjanje v primeru napake strojne opreme in opozarjanje uporabnika na novo meritev.

Podatki o uporabljani merilni in laboratorijski opremi se shranijo v nastavitveni pomnilnik programa. Tako je prihranjen čas pri postopku vnosa nove meritve.

Program omogoča *On-Line* in *Off-Line* delovanje strojne opreme.

Izvedeno je varnostno shranjevanje merjenih vrednosti.

Število odčitovalnikov, ki jih lahko program *MemoUpravljalnik* krmili, je 128, kolikor je tudi največje število odčitovalnikov, ki jih lahko krmili centralna krmilna enota sistema

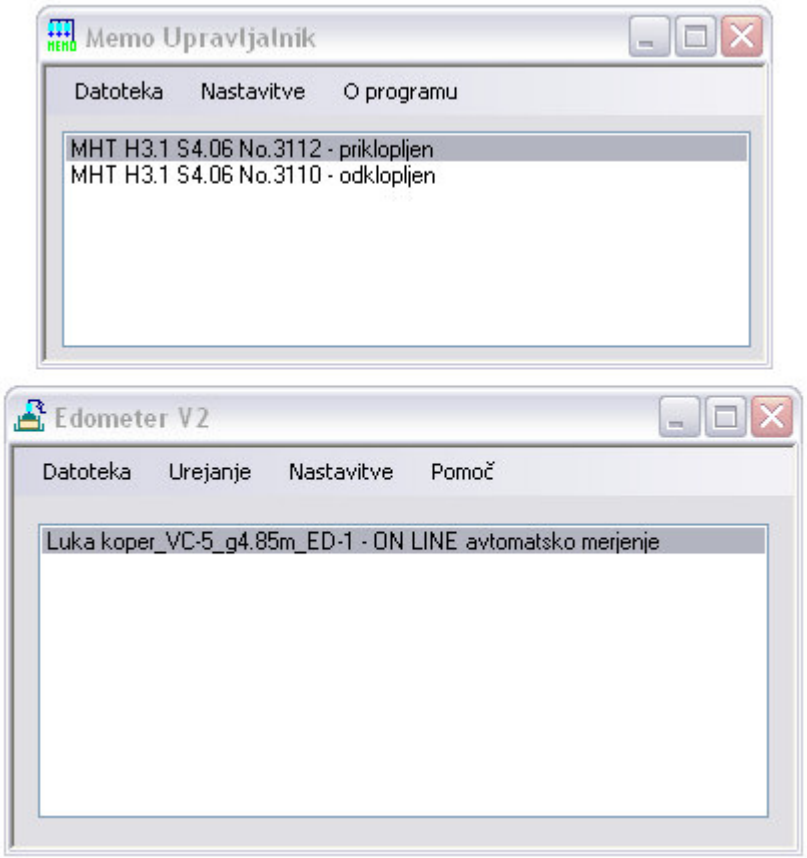
Oba programa imata okenski uporabniški vmesnik, delujeta na operacijskem sistemu Microsoft Windows XP in Vista, za delovanje potrebujeta izvajalno okolje CLR (Common Language Runtime), ki je vključeno v oba operacijska sistema.

6.2 Grobi prikaz uporabe

Sledeči primer uporabe merilnega sistema v KMTAL v ključnih korakih prikazuje celoten potek edometrske meritve.

V laboratoriju KMTAL se programa *Edometer V2* in *Memo Upravljalnik* trenutno izvajata na enem samem računalniku. Časovno se postavimo v točko pred začetkom nove edometrske meritve. V tem trenutku izvaja program *Edometer V2* obstoječo avtomatsko merjeno

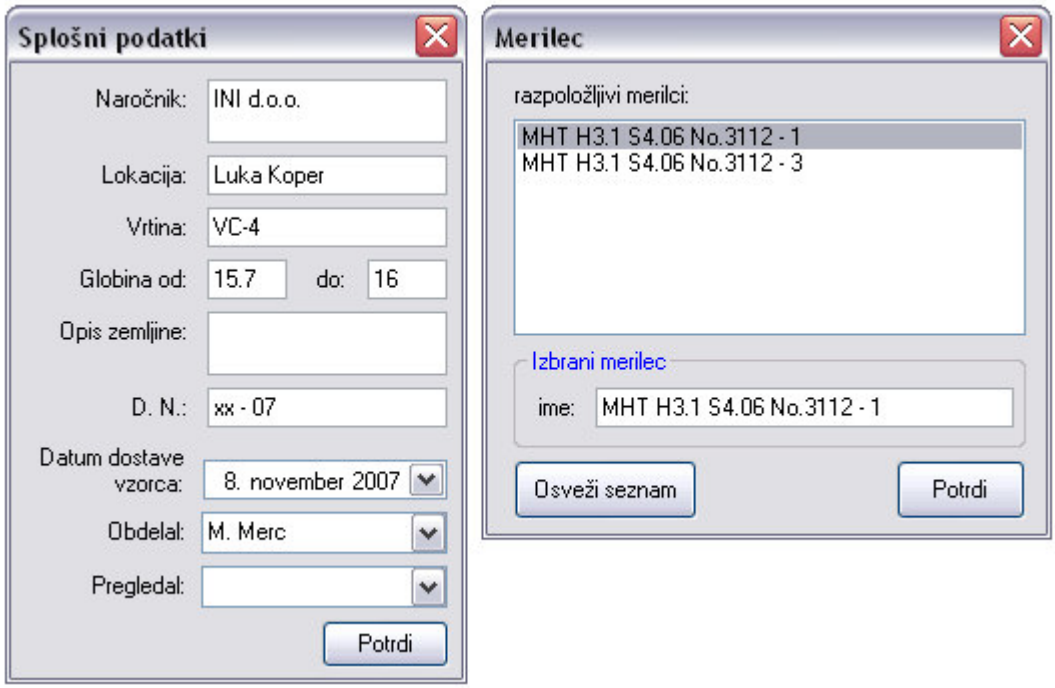
edometrsko meritev. Ker se oba programa nahajata na istem računalniku, na zaslonu vidimo okni obeh programov.



Slika 28: Okni programa Edometer V2 in Memo Upravljalnik

Slika 28 prikazuje okni obeh programov. Vidimo, da program *Edometer V2* trenutno izvaja eno samo edometrsko meritev. V oknu programa *Memo Upravljalnik* vidimo registrirana dva odčitovalnika, izmed katerih je trenutno vklopljen le prvi.

Novi vzorec vgradimo v edometrsko celico, celico namestimo na edometrski aparat in nastavimo merilnik pomika, da ustrezno nalega na merilno mesto aparata. V meniju *Urejanje* programa *Edometer V2* izberemo vnos nove avtomatsko vodene edometrske meritve. Program zahteva preko dialognih oken vnos osnovnih podatkov o edometrski meritvi. Če katerega podatka trenutno ne želimo določiti, lahko to storimo kasneje. Pričakovan je vnos splošnih podatkov merjenja, vnos pričakovanega poteka bremenskih stopenj in vnos parametrov edometrskega aparata. Avtomatsko vodena preiskava zahteva pogovorni izbor merilnega kanala, na katerega je priklopljen merilnik pomika edometrskega aparata.

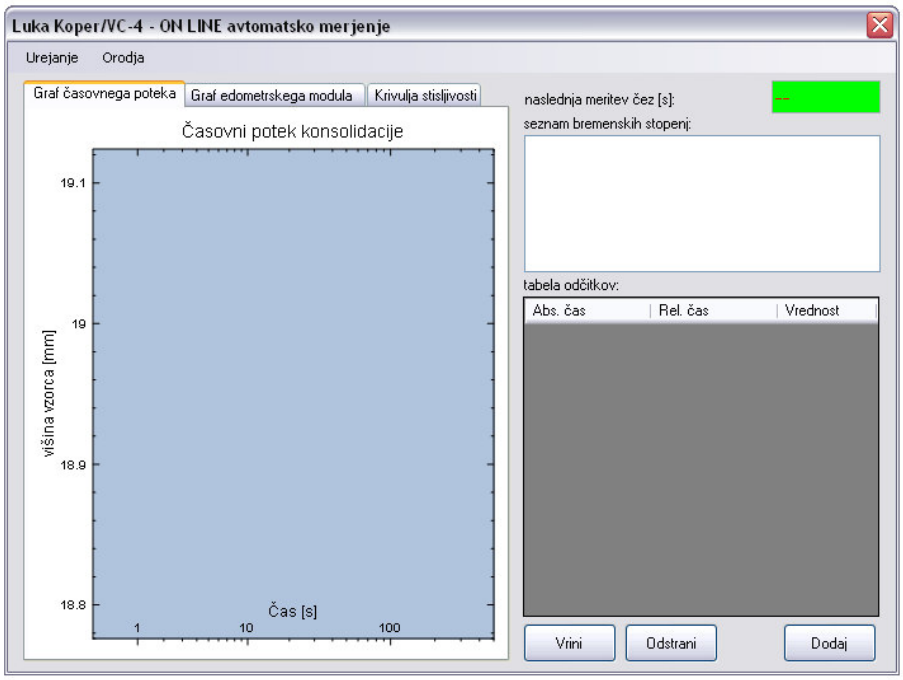


Slika 29: Dialogni okni za vnos splošnih podatkov o edometrski preiskavi in izbor kanala strojnega merjenja

Slika 29 prikazuje pogovorni okni za vnos splošnih podatkov o edometrski preiskavi in izbor kanala strojnega merjenja.

Prvo okno na sliki 29 je dialogno okno za vnos splošnih podatkov o edometrski preiskavi. Vnos podatkov ni obvezujoč, je pa priporočen. Drugo okno na sliki 29 je dialogno okno za izbor merilnega kanala strojnega merjenja. V seznamu razpoložljivih merilcev vidimo, da merilnega kanala 2 ni mogoče izbrati, ker je že zaseden, saj je ena preiskava že v teku. Ob pritisku na gumb *Potrdi* se izbrani merilni kanal izbranega odčitovalnika rezervira samo za to preiskavo.

Ko je vpisovanje zahtevanih podatkov o edometrski meritvi končano, se meritev doda v seznam meritev programa *Edometer V2*. Okno za upravljanje s posamezno meritvijo dobimo z dvoklikom na izbrano meritev v prikazanem seznamu meritev programa v osnovnem oknu. Dvoklik nad pravkar dodano meritvijo odpre naslednje okno:



Slika 30: Okno za urejanje edometrske meritve ob začetku preiskave

Na sliki 30 vidimo okno za urejanje edometrske preiskave takoj po njenem začetku. Okno vsebuje menijsko vrstico. Leva stran vsebuje izbirnik za prikazovanje različnih tipov grafov o poteku preiskave. Izbiramo lahko med grafom časovnega poteka, grafom edometrskega modula in grafom krivulje stisljivosti. Desna stran okna prikazuje seznam bremenskih stopenj, spodnja stran pa tabelo za urejanje odčitkov posamezne bremenske stopnje. Povsem zgoraj desno je prikazovalnik časa, ki bo pretekel do naslednjega odčitka.

Dodani meritvi še nismo dodali nobene bremenske stopnje. To storimo s klikom na ukaz *Dodaj bremensko stopnjo,* ki se nahaja v meniju *Urejanje*. Odpre se pogovorno okno za vnos podatkov o novi bremenski stopnji.

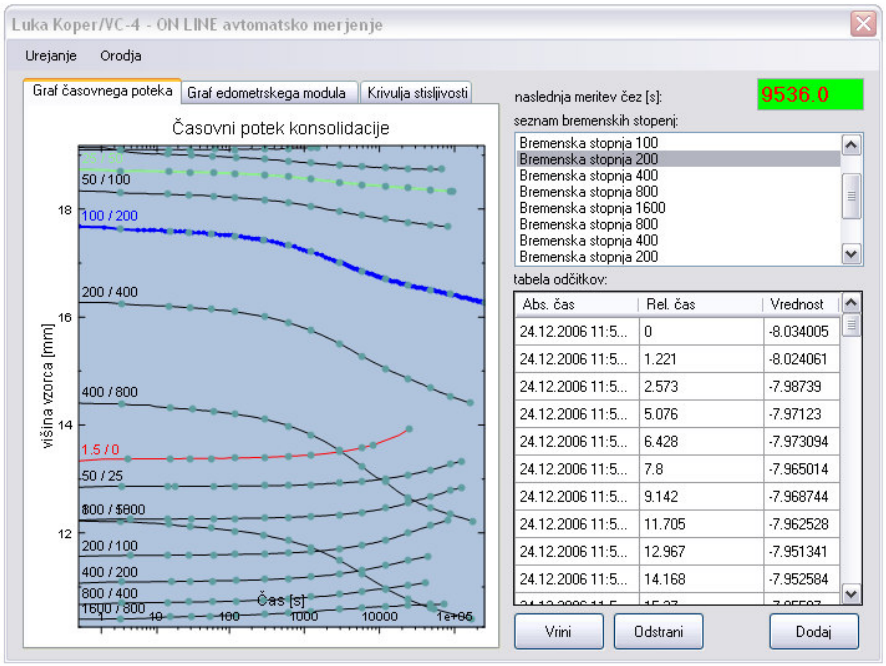
Slika 31: Pogovorno okno za vnos podatkov bremenske stopnje

V pogovornem oknu za vnos podatkov v prvo tekstovno polje vpišemo napetost nove bremenske stopnje. Program izračuna na podlagi prenosnega razmerja (lastnost edometrskega aparata) maso uteži, ki jih moramo dodati na obremenilni mehanizem. Ker vnašamo prvo bremensko stopnjo, je trenutna masa uteži 0. Za napetost 1.5 kPa moramo torej dodati 61 gramov uteži.

Nastavimo lahko tudi odsekoma podan časovni protokol merjenja v dodajani bremenski stopnji. V konkretnem primeru bo program v prvih 3 sekundah meril na interval 0 sekund. To pomeni, da se bo merjenje vršilo tako hitro, kolikor zmore merilni sistem, v našem primeru na slabo sekundo. V intervalu 3 do 15 sekund bo merjenje potekalo na vsaki 2 sekundi, itd.

S klikom na gumb *Potrdi* sprožimo odštevanje časa, ko moramo spustiti dodano utež na obremenilni mehanizem. Ob pritisku na gumb nas na odštevanje časa opozori tudi zvočni signal. Takoj po preteku časa, ob pisku previdno spustimo utež in bremenska stopnja se tako začne. Potek merjenja spremljamo v realnem času, vsaka novo izmerjena točka osveži prikaz. Postopek dodajanja bremenskih stopenj ponavljamo tako dolgo, dokler meritev ni končana.

Tik pred koncem obravnavane meritve je bil njen izgled v oknu za urejanje sledeč:



Slika 32: Pogled na okno za urejanje edometrske meritve pred koncem preiskave

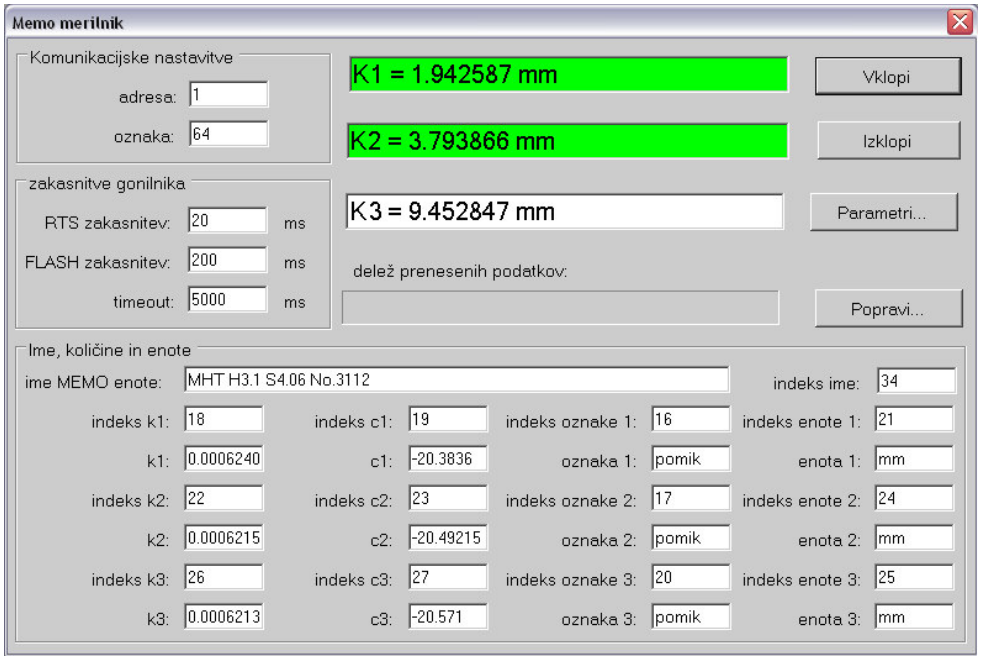
Slika 32 prikazuje okno za urejanje edometrske meritve malo pred koncem preiskave. Črta zadnje bremenske stopnje je rdeča. Bremenska stopnja, katere izmerjene vrednosti lahko urejamo v tabeli na desni strani, je označena z modro barvo. Z zeleno barvo je narisana črta bremenske stopnje, ki bo naslednja izbrana v urejanje ob kliku miške v njeno bližino.

Vsi grafi imajo svoje kontekstne menije, v katerih lahko izbiramo različne tipe meril osi; linearno, korensko in logaritemsko. Preko kontekstnih menijev posameznega grafa sprožimo izvajanje različnih grafičnih postopkov za izračun dodatnih parametrov konsolidacije.

Ko želimo meritev zaključiti, nas program ponovno preko dialognih oken vodi k opcijskemu vnosu rezultatov tehtanja vzorca in imena datoteke, kamor se shrani celotna meritev. Edometrska meritev je s tem zaključena.

Kočni rezultat edometrske preiskave je v standarni obliki izdelano poročilo običajno izpisano na papir. Program *Edometer V2* funkcionalnosti za oblikovanje in izpis končnega poročila nima. V laboratoriju KMTAL je za ta namen uporabljen program *Microsoft Excel*. Znotraj programa Excel je bil v programskem jeziku VBA dodatno izdelan modul, ki zna na klik miške prebrati datoteko rezultatov programa *Edometer V2* in rezultate oblikovati ter izpisati v končni obliki. Rezultat izveden s programom Excel je prikazan v prilogi 2.

Upravljalni program *Memo Upravljalnik* služi predvsem prikazu tekočega stanja posameznega odčitovalnika merilnega sistema.



Slika 33: Pogovorno okno za prikaz parametrov Memo odčitovalnika

Dialogno okno za prikaz parametrov Memo odčitovalnika omogoča prikaz trenutnih merjenih vrednosti na vsakem izmed treh merilnih kanalov. Omogoča tudi nastavitev posameznih parametrov odčitovalnika; faktorja in konstante merilnega kanala, enot in merjenih količin. Zelena barva polja prikaza vrednosti pomeni, da je merilni kanal v *On-Line* uporabi. Merilni kanal je lahko tudi v *Off-Line* uporabi. Takrat je barva polja temno zelena.

7 ZAKLJUČEK

Avtomatiziran merilni sistem Edometer V2/Memo Upravljalnik je bil v laboratoriju KMTAL brez težav uporabljen pri nekaj 10 primerih. Pri razvoju obeh programov so z idejami in koristnimi predlogi sodelovali tudi laboranti laboratorija KMTAL.

Informacijska tehnologija je z razvojem zelo zmogljivih in dostopnih računalnikov dosegla zelo velik napredek, ki je opazen tudi na področju razvoja programske opreme.

Nekdanje okorne programske jezike, proceduralni način programiranja in zapletena razvojna orodja so zamenjali zmogljivi objektno usmerjeni programski jeziki. Razvojna orodja so postala vizualna. Zaradi uporabe grafike, novih konceptov in visoke stopnje avtomatizacije pisanja programske kode se je tudi njen obseg precej zmanjšal. Največ je k temu prispeval ravno objektni način programiranja.

Kompleksni, matematično zahtevni in težko rešljivi problemi so danes z ustrezno programsko opremo rešeni v delčku sekunde. Brezžična komunikacija na vseh nivojih bo tudi v prihodnjih letih imela velik vpliv na naše življenje.

Ko ob vsem razvoju pogledamo način izvedbe meritev v geomehanskih laboratorijih, dobimo občutek, da na področju uporabe računalniških tehnologij zaostaja za možnostmi. Verjetno tiči razlog v majhnosti tržišča in v tem, da se sami procesi merjenja zaradi narave zemljin odvijajo počasi in posledično generirajo relativno majhno količino podatkov potrebnih dodatne obdelave.

Edometrsko preiskavo lahko še danes z nekaj potrpežljivosti izvajamo ročno in poročilo izdelamo z ročnim vnosom podatkov v program Excel. Če bi bilo podatkov več, bi se slednja že v preteklosti bolj avtomatizirala.

Pri pregledu svetovnega spleta edometrskih merilnih sistemov podobnih sistemu *Edometer V2 / Memo Upravljalnik* v času izdelave te diplome ni bilo opaziti. Z malo truda pri določenih izboljšavah in predvsem podpori še kakšnemu drugemu sistemu zajema podatkov (proizvajalec merilnih sistemov ELE) bi sistem utegnil postati tudi tržno zanimiv.

Na tržišču obstajajo tudi edometrski aparati z avtomatskim obremenjevanjem vzorca. Krmilna zasnova razvitega sistema je prilagojena tudi tovrstnim rešitvam, vendar v okviru diplomskega dela ni bila izvedena. Avtomatsko obremenjevanje vzorca bi sistem popolnoma avtomatiziralo. Vzorec bi tako lahko kontrolirano ciklično obremenjevali in razbremenjevali, kar bi omogočilo nove vrste preiskav.

8 VIRI

1. Casagrande, A., 1932. The structure of clay and its importance in foundation engineering. J. Boston Soc. Civ. Eng., 19, No. 4, str. 168-182
2. Casagrande, A., 1936. The determination of the pre-consolidation load and its practical significance. Proc. 1st Int. Conf. Soil Mech., Cambridge, Mass., Vol. 3
3. Fay, T. M., Selfon S., Fay, T. J. 2003. DirectX Audio Exposed: Interactive Audio Development, Wordware Publishing: 550 str.
4. Liberty, J. 2005. Programming C#: Building .NET Applications, 4.izdaja, USA, O'Reilly Media: 666 str.
5. Lowy, J. 2005. Programming .NET Components, 2. izdaja, USA, O'Reilly Media: 644 str.
6. Taylor, D. W., 1942. Research on consolidation clays. M.I.T., Dept. Of Civ. and Sanit. Eng., No. 82
7. Terzaghi, K., 1943. Theoretical Soil Mechanics. New York, Wiley

Standardi:

1. ISO/TS 17892-5:2004 Geotechnical investigation and testing - Laboratory testing of soil - Part 5: Incremental loading oedometer test
2. W3C XML 1.0:2006 Extensible Markup Language (XML) 1.0 (Fourth Edition)

PRILOGE

Priloga 1: Primer izvorne kode razreda BremenskaStopnja

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;
using System.IO;
using System.Collections;

namespace EdometerV2
{
    /// <summary>
    /// Razred 'BremenskaStopnja' določa podatke o bremenski stopnji.
    /// </summary>
    public class BremenskaStopnja
    {
        #region Spremenljivke

        /// <summary>
        /// Edometrska meritev, kateri pripada bremenska stopnja.
        /// </summary>
        private EdometrskaMeritev meritev;
        /// <summary>
        /// Napetost bremenske stopnje.
        /// </summary>
        private float napetost;
        /// <summary>
        /// Seznam točk.
        /// </summary>
        private ArrayList točke;
        /// <summary>
        /// Gradnik za Taylorjev postopek.
        /// </summary>
        private Taylor taylor;
        /// <summary>
        /// Časovni protokol merjenja.
        /// </summary>
        private ČasovniProtokol časovniProtokol;
        /// <summary>
        /// Izračun sekundarne konsolidacije.
        /// </summary>
        private SekundarnaKonsolidacija sekundarnaKonsolidacija;

        #endregion

        #region Lastnosti

        /// <summary>
        /// Časovni protokol merjenja.
        /// </summary>
        public ČasovniProtokol ČasovniProtokol
        {
            get
            {
```

```
            return časovniProtokol;
        }
        set
        {
            časovniProtokol = value;
        }
    }

    /// <summary>
    /// Edometrska meritev, kateri pripada bremenska stopnja.
    /// </summary>
    public EdometrskaMeritev Meritev
    {
        get
        {
            return meritev;
        }
    }

    /// <summary>
    /// Napetost bremenske stopnje.
    /// </summary>
    public float Napetost
    {
        get
        {
            return napetost;
        }
        set
        {
            napetost = value;
        }
    }

    /// <summary>
    /// Število točk bremenske stopnje.
    /// </summary>
    public int ŠteviloTočk
    {
        get
        {
            return točke.Count;
        }
    }

    /// <summary>
    /// Gradnik za Taylorjev postopek izračuna konca konsolidacije.
    /// </summary>
    public Taylor Taylor
    {
        get
        {
            return taylor;
        }
    }

    /// <summary>
    /// Izračun sekundarne konsolidacije.
    /// </summary>
    public SekundarnaKonsolidacija SekundarnaKonsolidacija
```

```
{
    get
    {
        return sekundarnaKonsolidacija;
    }
}

#endregion

#region Konstruktorji

/// <summary>
/// Konstruktor.
/// </summary>
/// <param name="meritev">Meritev, kateri pripada bremenska
/// stopnja.</param>
public BremenskaStopnja(EdometrskaMeritev meritev)
{
    this.meritev = meritev;
    točke = new ArrayList();
    taylor = new Taylor(this);
    sekundarnaKonsolidacija = new SekundarnaKonsolidacija(this);
    časovniProtokol = new ČasovniProtokol();
}

/// <summary>
/// Konstruktor.
/// </summary>
/// <param name="element">Xml element s podatki.</param>
/// <param name="meritev">Meritev, kateri pripada bremenska
/// stopnja.</param>
public BremenskaStopnja(XmlElement element,
    EdometrskaMeritev meritev)
{
    if (element.Name != "BremenskaStopnja")
        throw new Exception("Neveljavno ime elementa s podatki" +
            "bremenske stopnje!");
    this.meritev = meritev;
    napetost = BesedaŠtevilka.BesedaFloat(
        element.GetAttribute("napetost"));
    časovniProtokol = new ČasovniProtokol(
        (XmlElement)element.SelectSingleNode("ČasovniProtokol"));
    XmlNodeList seznamTočk = element.SelectNodes("Točka");
    int n = seznamTočk.Count;
    Točka točka;
    točke = new ArrayList(n);
    if (element.HasAttribute("časZačetka"))
    {
        DateTime časZačetka = BesedaŠtevilka.BesedaČas(
            element.GetAttribute("časZačetka"));
        float sekunde;
        for (int i = 0; i < n; i++)
        {
            točka = new Točka((XmlElement)seznamTočk[i], this);
            sekunde = (float)(točka.AbsolutniČas.Subtract(
                new DateTime()).TotalSeconds);
            točka.AbsolutniČas = časZačetka.AddSeconds(sekunde);
            točke.Add(točka);
        }
    }
```

```
        else
        {
            for (int i = 0; i < n; i++)
            {
                točka = new Točka((XmlElement)seznamTočk[i], this);
                točke.Add(točka);
            }
        }
        taylor = new Taylor(
            (XmlElement)element.SelectSingleNode("Taylor"), this);
        XmlElement sekKonsXml = (XmlElement)
            element.SelectSingleNode("SekundarnaKonsolidacija");
        if (sekKonsXml != null)
            sekundarnaKonsolidacija = new SekundarnaKonsolidacija(
                sekKonsXml, this);
        else
            sekundarnaKonsolidacija = new SekundarnaKonsolidacija(
                this);
    }

    #endregion

    #region Zasebne metode

    #endregion

    #region Javne metode

    /// <summary>
    /// Vrne izbrano točko.
    /// </summary>
    /// <param name="indeksTočke">Indeks izbrane točke.</param>
    /// <returns>Vrne izbrano točko.</returns>
    public Točka Točka(int indeksTočke)
    {
        return (Točka)točke[indeksTočke];
    }

    /// <summary>
    /// Doda točko v seznam.
    /// </summary>
    /// <param name="točka">Točka, ki se doda v seznam.
    /// Mora pripadati tej stopnji!</param>
    public void DodajTočko(Točka točka)
    {
        if (točka.BremenskaStopnja != this)
            throw new Exception("Točka ne pripada tej " +
                "bremenski stopnji!");
        meritev.SkupniPodatki.Mutex.WaitOne();
        točke.Add(točka);
        meritev.SkupniPodatki.Mutex.ReleaseMutex();
        if (meritev.IgralnikNovaTočka != null)
            meritev.IgralnikNovaTočka.Igraj();
        switch (meritev.GetType().Name)
        {
            case "RočnaEdometrskaMeritev":
                RočnaEdometrskaMeritev rem =
                    (RočnaEdometrskaMeritev)meritev;
                if(rem.Dodajanje)
                    rem.PotrdiDodajanje();
```

```
                break;
            case "AvtomatskaEdometrskaMeritev":
                break;
            default:
                throw new Exception("Neimplementiran tip " +
                    "edometrske meritve!");
        }
    }

    /// <summary>
    /// Doda novo točko v seznam točk.
    /// </summary>
    /// <param name="absolutniČas">Absolutni čas točke.</param>
    /// <param name="odčitek">Odčitek točke.</param>
    public void DodajTočko(DateTime absolutniČas, float odčitek)
    {
        meritev.SkupniPodatki.Mutex.WaitOne();
        točke.Add(new Točka(this, absolutniČas, odčitek));
        meritev.SkupniPodatki.Mutex.ReleaseMutex();
        if (meritev.IgralnikNovaTočka != null)
            meritev.IgralnikNovaTočka.Igraj();
        switch (meritev.GetType().Name)
        {
            case "RočnaEdometrskaMeritev":
                RočnaEdometrskaMeritev rem =
                    (RočnaEdometrskaMeritev)meritev;
                rem.PotrdiDodajanje();
                break;
            case "AvtomatskaEdometrskaMeritev":
                break;
            default:
                throw new Exception("Neimplementiran tip " +
                    "edometrske meritve!");
        }
    }

    /// <summary>
    /// Doda polje točk v seznam točk.
    /// </summary>
    /// <param name="absolutniČasi">Polje absolutnih časov.</param>
    /// <param name="odčitki">Polje odčitkov.</param>
    public void DodajTočke(DateTime[] absolutniČasi, float[] odčitki)
    {
        int n = absolutniČasi.Length;
        if (n != odčitki.Length)
            throw new Exception("Polja časov in odčitkov " +
                "nista enako dolga!");
        meritev.SkupniPodatki.Mutex.WaitOne();
        for (int i = 0; i < n; i++)
            točke.Add(new Točka(this, absolutniČasi[i], odčitki[i]));
        meritev.SkupniPodatki.Mutex.ReleaseMutex();
    }

    /// <summary>
    /// Vrine točko na izbrano mesto v seznamu.
    /// </summary>
    /// <param name="indeksTočke">Indeks na katerega se
    /// točka vrine.</param>
    /// <param name="absolutniČas">Absolutni čas točke.</param>
    /// <param name="odčitek">Odčitek točke.</param>
```

```
public void VriniTočko(int indeksTočke, DateTime absolutniČas,
    float odčitek)
{
    meritev.SkupniPodatki.Mutex.WaitOne();
    točke.Insert(indeksTočke,
        new Točka(this, absolutniČas, odčitek));
    meritev.SkupniPodatki.Mutex.ReleaseMutex();
}
/// <summary>
/// Odstrani izbrano točko.
/// </summary>
/// <param name="indeks">Indeks izbrane točke.</param>
/// <returns>Vrne odstranjeno točko.</returns>
public Točka OdstraniTočko(int indeks)
{
    meritev.SkupniPodatki.Mutex.WaitOne();
    Točka t = Točka(indeks);
    točke.RemoveAt(indeks);
    t.Konec();
    meritev.SkupniPodatki.Mutex.ReleaseMutex();
    return t;
}
/// <summary>
/// Sprosti referenco na edometrsko meritev.
/// </summary>
public void Konec()
{
    meritev = null;
}
/// <summary>
/// Shrani podatke o bremenski stopnji.
/// </summary>
/// <param name="element">Xml element s podatki o bremenski
/// stopnji.</param>
public void Shrani(XmlElement element)
{
    if (element.Name != "BremenskaStopnja")
        throw new Exception("Neveljavno ime elementa s " +
            "podatki bremenske stopnje!");
    element.SetAttribute("napetost", napetost.ToString());
    XmlElement časovniProtokolXml =
        element.OwnerDocument.CreateElement("ČasovniProtokol");
    časovniProtokol.Shrani(časovniProtokolXml);
    element.AppendChild(časovniProtokolXml);
    XmlElement točkaXml;
    for (int i = 0; i < ŠteviloTočk; i++)
    {
        točkaXml = element.OwnerDocument.CreateElement("Točka");
        Točka(i).Shrani(točkaXml);
        element.AppendChild(točkaXml);
    }
    XmlElement taylorXml =
        element.OwnerDocument.CreateElement("Taylor");
    taylor.Shrani(taylorXml);
    element.AppendChild(taylorXml);
    XmlElement konsolidacijaXml =
        element.OwnerDocument.CreateElement(
            "SekundarnaKonsolidacija");
    sekundarnaKonsolidacija.Shrani(konsolidacijaXml);
    element.AppendChild(konsolidacijaXml);
```

```
        }
        /// <summary>
        /// Shrani podatke o bremenski stopnji.
        /// </summary>
        /// <param name="element">Xml element s podatki o
        /// bremenski stopnji.</param>
        /// <param name="absolutniČas">Določa, ali naj bo kot podatek časa
        /// absolutni čas ali relativni čas.</param>
        public void Shrani(XmlElement element, bool absolutniČas)
        {
            if (absolutniČas)
                Shrani(element);
            else
            {
                if (element.Name != "BremenskaStopnja")
                    throw new Exception("Neveljavno ime elementa " +
                        "s podatki bremenske stopnje!");
                element.SetAttribute("napetost", napetost.ToString());
                XmlElement časovniProtokolXml =
                    element.OwnerDocument.CreateElement("ČasovniProtokol");
                časovniProtokol.Shrani(časovniProtokolXml);
                element.AppendChild(časovniProtokolXml);
                XmlElement točkaXml;
                if (ŠteviloTočk > 0)
                {
                    element.SetAttribute("časZačetka",
                        BesedaŠtevilka.ČasBeseda(Točka(0).AbsolutniČas));
                    for (int i = 0; i < ŠteviloTočk; i++)
                    {
                        točkaXml =
                            element.OwnerDocument.CreateElement("Točka");
                        Točka(i).Shrani(točkaXml, false);
                        element.AppendChild(točkaXml);
                    }
                }
                XmlElement taylorXml =
                    element.OwnerDocument.CreateElement("Taylor");
                taylor.Shrani(taylorXml);
                element.AppendChild(taylorXml);
                XmlElement konsolidacijaXml =
                    element.OwnerDocument.CreateElement(
                        "SekundarnaKonsolidacija");
                sekundarnaKonsolidacija.Shrani(konsolidacijaXml);
                element.AppendChild(konsolidacijaXml);
            }
        }
        /// <summary>
        /// Nastavi meritev bremenski stopnji.
        /// </summary>
        /// <param name="meritev">Edometrska meritev.</param>
        public void NastaviMeritev(EdometrskaMeritev meritev)
        {
            this.meritev = meritev;
        }

        #endregion
    }
}
```

Priloga 2: Primer končnega poročila edometrske preiskave

Univerza v Ljubljani
Fakulteta za gradbeništvo in geodezijo

Katedra za mehaniko tal z laboratorijem

Jamova c. 2, p.p.3422
1000 Ljubljana, Slovenija
telefon 01 4768 500
faks 01 4250 681

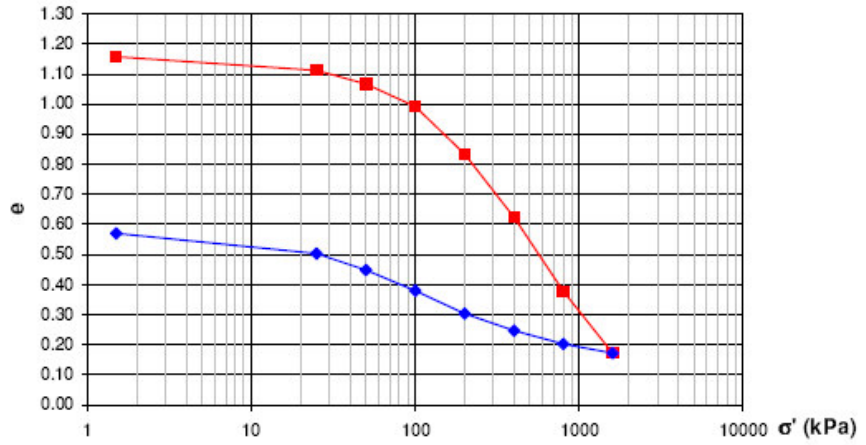
EDOMETERSKI PRESKUS S POSTOPNIM OBREMENJEVANJEM

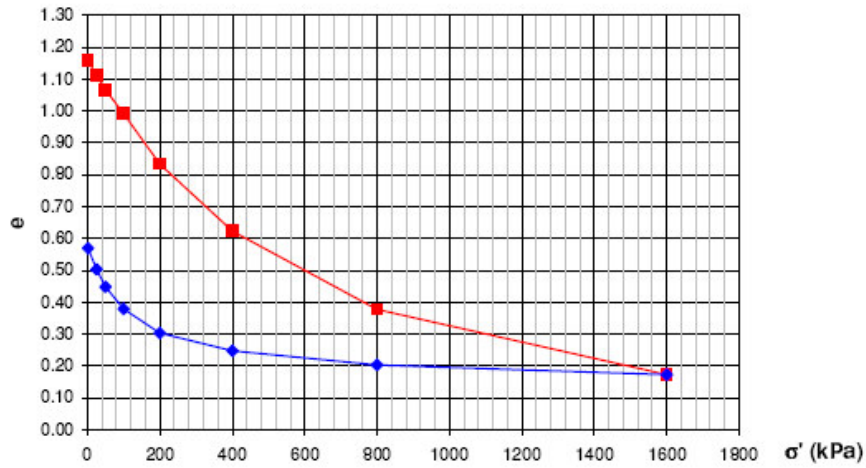
SIST/ISO/TS 17892-5:2004

LOKACIJA:	2-Luka Koper	**D.N.:**	42-06
VRTINA:	VC-4	**DATUM DOSTAVE:**	10.11.06
GLOBINA:	15.7-16 m	**POROČILO:**	

Aparat:	1		gostota zrn ρ_s:	2.75	t/m^3
višina vzorca:	19.1	mm	vlaga vzorca pred preiskavo:	43.6	%
premer vzorca:	70.0	mm	vlaga vzorca po preiskavi:	35.5	%
			gostota ρ:	1.83	t/m^3
			suha gostota ρ_d:	1.28	t/m^3

KRIVULJA STISLJIVOSTI





PREISKAL:	M. Merc	**PREGLEDAL:**	dr. A. Petkovšek
ZAČ. PREISKAVE:	21.12.06		
KON. PREISKAVE:	08.01.07	**PRILOGA:**	

Univerza v Ljubljani
Fakulteta za gradbeništvo in geodezijo

Katedra za mehaniko tal z laboratorijem

Jamova c. 2,p.p.3422
1000 Ljubljana, Slovenija
telefon 01 4768 500
faks 01 4250 681

EDOMETERSKI PRESKUS S POSTOPNIM OBREMENJEVANJEM

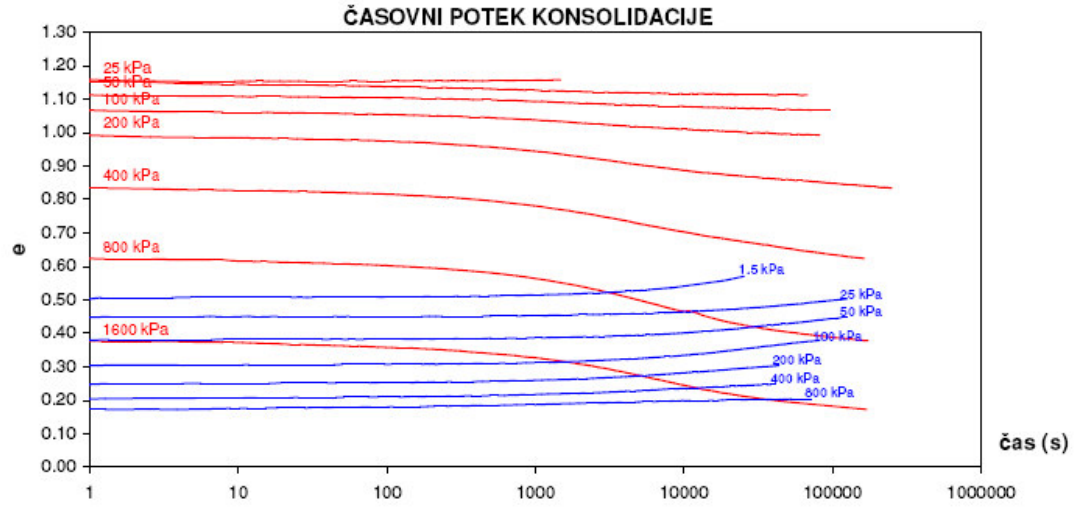
SIST/ISO/TS 17892-5:2004

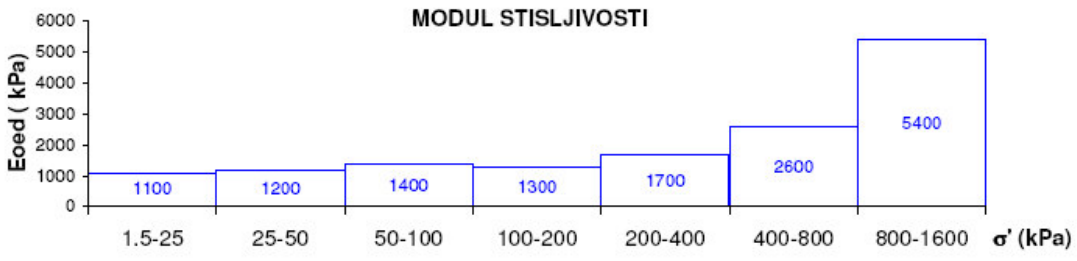
LOKACIJA: 2-Luka Koper
VRTINA: VC-4
GLOBINA: 15.7-16 m

D.N.: 42-06
DATUM DOSTAVE: 10.11.06
POROČILO:

stopnja (kPa)	E (kPa)	c_v (m^2/s)	k(c_v) (m/s)	Cα
1.5-25	1100			
25-50	1200	0.00E+00	0.00E+00	0.0000
50-100	1400	0.00E+00	0.00E+00	0.0000
100-200	1300	1.09E-08	8.63E-11	-0.0242
200-400	1700	9.03E-10	5.18E-12	-0.0349
400-800	2600			
800-1600	5400	0.00E+00	0.00E+00	
1600-800	30000	0.00E+00	0.00E+00	
800-400	11000	0.00E+00	0.00E+00	
400-200	4400			
200-100	1700			
100-50	1000			
50-25	660			
25-1.5	530	0.00E+00	0.00E+00	0.0000

σ'_p (kPa)	99.3
C_c	0.74
C_s	0.00
λ	0.32
κ	0.00





PREISKAL: M. Merc
ZAČ. PREISKAVE: 21.12.06
KON. PREISKAVE: 08.01.07

PREGLEDAL: dr. A. Petkovšek
PRILOGA: