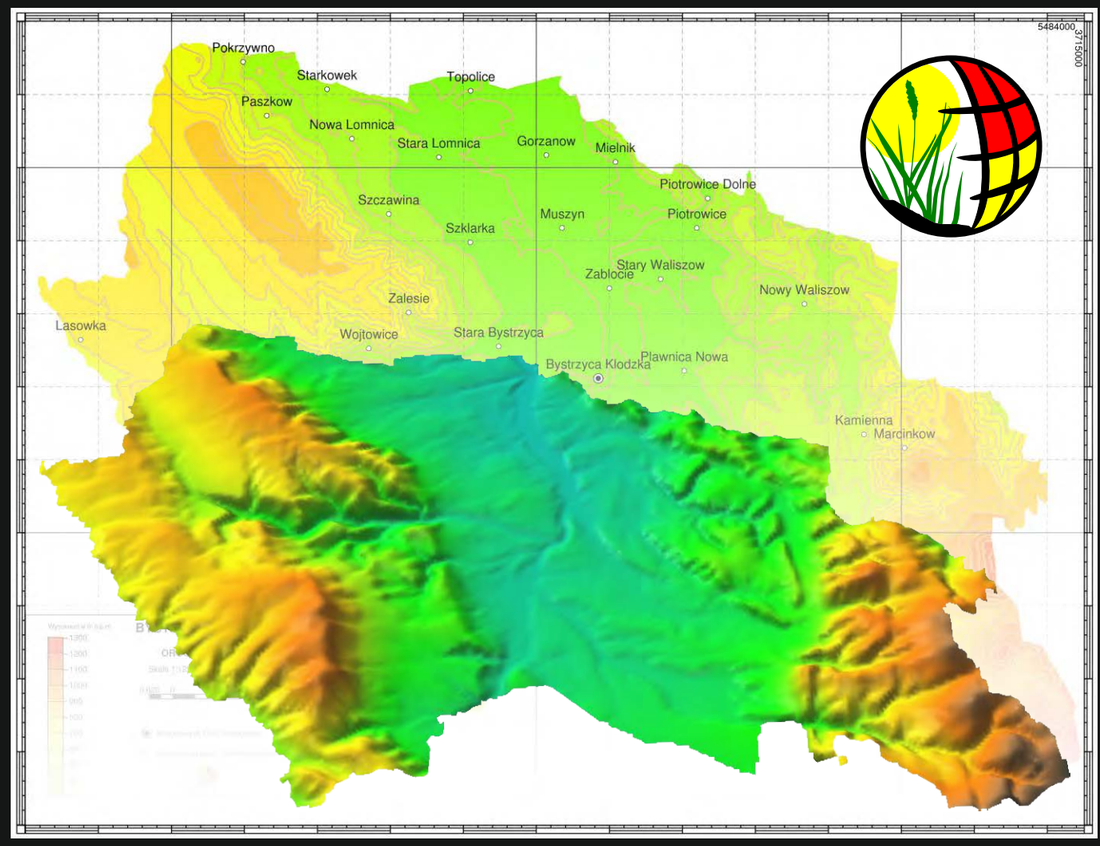
Rozprawy Naukowe Instytutu Geografii i Rozwoju Regionalnego Uniwersytetu Wrocławskiego 15

Uniwersytet Wrocławski

ANALIZY PRZESTRZENNE Z WYKORZYSTANIEM GRASS

pod redakcją PAWŁA NETZELA



ISBN 978-83-62673-02-5

Wrocław 2011

Analizy przestrzenne z wykorzystaniem GRASS

Rozprawy Naukowe Instytutu Geografii i Rozwoju Regionalnego Uniwersytetu Wrocławskiego 15

Analizy przestrzenne z wykorzystaniem GRASS

pod redakcją Pawła Netzela

Instytutu Geografii i Rozwoju Regionalnego Uniwersytetu Wrocławskiego
Wrocław 2011

Analizy przestrzenne z wykorzystaniem GRASS

Autorzy:

Adam Górecki
Dolnośląski Ośrodek Badawczy we Wrocławiu, Instytut Technologiczno-Przyrodniczy, Falenty

Paweł Netzel (red.)
Zakład Klimatologii i Ochrony Atmosfery, Instytutu Geografii i Rozwoju Regionalnego,
Uniwersytet Wrocławski,
Fundacja OSGeo Oddział Polski,
Wrocławska Grupa Użytkowników GRASS

Milena Nowotarska
Fundacja OSGeo Oddział Polski

Hanna Ojrzyńska
Zakład Klimatologii i Ochrony Atmosfery, Instytutu Geografii i Rozwoju Regionalnego,
Uniwersytet Wrocławski

Małgorzata Pietras
Zakład Klimatologii i Ochrony Atmosfery, Instytutu Geografii i Rozwoju Regionalnego,
Uniwersytet Wrocławski

Robert Szczepanek
Politechnika Krakowska, Fundacja OSGeo Oddział Polski

Jacek Ślopek
Wrocławska Grupa Użytkowników GRASS

Instytutu Geografii i Rozwoju Regionalnego Uniwersytetu Wrocławskiego
Wrocław 2011

Rozprawy Naukowe Instytutu Geografii i Rozwoju Regionalnego 15

Redaktor serii
Zdzisław Jary

Recenzenci tomu
Maciej Kryza
Tomasz Kubik
Krzysztof Migała

Skład komputerowy
Paweł Netzel

Projekt graficzny okładki
Waldemar Spallek, Marek Kasprzak

Zalecane cytowanie
Netzel P. (red.), 2011. Analizy przestrzenne z wykorzystaniem GRASS.
Rozprawy Naukowe Instytutu Geografii i Rozwoju Regionalnego 15,
Uniwersytet Wrocławski, Wrocław.

©Copyright 2011 by Instytut Geografii i Rozwoju Regionalnego Uniwersytetu Wrocławskiego

ISBN 978-83-62673-02-5

Instytut Geografii i Rozwoju Regionalnego
Plac Uniwersytecki 1, 50-137 Wrocław

Uniwersytet Wrocławski

Druk i oprawa
I-BiS s.c., ul. Lelewela 4, 53-505 Wrocław

Spis treści

1. Wstęp

Paweł Netzel

Oprogramowanie FOSS (ang. *Free Open Source Software*) zdobywa coraz większą popularność. Mówi się już o nowym modelu biznesowym, w którym firmy wspierające tworzenie takiego oprogramowania odnoszą zyski nie ze sprzedaży pakietów programowych a z implementacji i udzielania wsparcia dla oprogramowania FOSS. Nie ominęło to i takiej dziedziny oprogramowania jaką jest GIS. Jednym z programów FOSS dedykowanym do analiz z zakresu GIS jest program GRASS.

GRASS jest wykorzystywany na Uniwersytecie Wrocławskim do analiz przestrzennych od ponad 14 lat. Wykorzystywany jest zarówno do prostych obliczeń wykonywanych na danych przestrzennych, jak również do modelowania. Służy on także jako system składujący dane przestrzenne i pośredniczący w wymianie informacji pomiędzy innymi programami. System GRASS znalazł też swoje miejsce w dydaktyce w Zakładzie Klimatologii i Ochrony Atmosfery. Z jego wykorzystaniem jest rozwijana dydaktyka, powstają prace magisterskie i rozprawy doktorskie.

Od samego początku pracy z systemem pojawiła się chęć dzielenia się wiedzą i umiejętnościami wykorzystania GRASS do rozwiązywania problemów związanych z analizami przestrzennymi. Ideę pracy w systemie GRASS zaszczepił w Zakładzie profesor Friedrich Quiel z Królewskiej Politechniki Sztokholmskiej. Dzielił się on z nami własną wiedzą i znajomością systemu oraz metod analizy przestrzennej zgodnie z ideami przyświecającymi oprogramowaniu FOSS. Tak zrodził się pomysł warsztatów poświęconych zarówno analizom przestrzennym, jak i wykorzystaniu systemu GRASS.

W roku 2010 mieliśmy przyjemność organizować XII edycję warsztatów. Głównym organizatorem był Uniwersytet Wrocławski a w szczególności Pracownia Metod Modelowania Przestrzennego Środowiska Geograficznego będąca w strukturze Zakładu Klimatologii i Ochrony Atmosfery, Instytutu Geografii i Rozwoju Regionalnego oraz Wrocławska Grupa Użytkowników GRASS. Patronatem swoim objęła warsztaty fundacja OSGeo Oddział Polski.

Jak zwykle uczestnicy warsztatów należeli zarówno do grupy początkującej, jak i zaawansowanej. Osobami prowadzącymi byli zarówno pracownicy Uniwersytetu, doktoranci dzielący się swoimi doświadczeniami, jak też goście z ośrodka krakowskiego, członkowie PSC (ang. *Project Steering Commitee*) GRASS, członkowie OSGeo Poland, członkowie Wrocławskiej Grupy Użytkowników GRASS.

W ramach warsztatów prowadzone były wykłady, prezentacje oraz zajęcia praktyczne. Tradycyjnie dla osób zaczynających swoją przygodę z GRASS zorganizowano zajęcia wprowadzające do systemu GRASS.

Część wykładów i prezentacji znalazła swoje odzwierciedlenie w przygotowanych po warsztatach artykułach. Artykuły te poruszają różne aspekty wykorzystania systemu GRASS. Można wśród nich znaleźć:

- informacje o samym systemie, społeczności użytkowników, możliwości udziału w tworzeniu i rozwijaniu systemu,
- przykłady wykorzystania GRASS do konkretnych analiz,
- przykład rozwijania systemu o nowe metody analiz,
- opisy interfejsu użytkownika, jak i przedstawiania wyników analiz w postaci map,
- zasady łączenia systemu GRASS z pakietem statystycznym R.

W treści artykułów zamieszczone zostały pełne wersje skryptów oraz opisana składnia, wykorzystanych do analiz, poleceń systemu GRASS. W części artykułów wykorzystano dane przestrzenne dla gminy Bystrzyca Kłodzka. Dane te zostały udostępniony na potrzeby Warsztatów przez Adama Góreckiego z Dolnośląskiego Ośrodka Badawczego Instytutu Technologiczno-Przyrodniczego w Falentach.

W imieniu organizatorów warsztatów mam nadzieję, że prezentowane treści będą przydatne dla osób już wykorzystujących GRASS. Mam też nadzieję że będą one pomocne osobom, które dopiero szukają wygodnego i wolnodostępnego systemu do analiz przestrzennych.

2. Społeczność i polonizacja GRASS

Milena Nowotarska

2.1. Wstęp

GRASS jest projektem open source [1, 4], rozwijającym się dzięki społeczności skupionej wokół niego. Społeczność ta składa się z programistów, użytkowników, tłumaczy, osób piszących dokumentację, ale to nie jest podział; te role się przenikają. Kto lepiej napisze dokumentację, niż autor danej funkcji? Następnie taką dokumentację może uzupełniać każdy. To jest właśnie wolność, którą dają swoim użytkownikom projekty oprogramowania wolnego, o otwartym kodzie źródłowym.

Kto chętniej zrobi tłumaczenie interfejsu programu, niż jego użytkownik lub nauczyciel? Do tego nie trzeba posiadać umiejętności programowania. Narzędzia do tłumaczenia są dostępne od ręki, także na wolnych licencjach, źródła do tłumaczeń też.

2.2. Historia społeczności GRASS

Początki systemu GRASS wywodzą się ze wczesnych lat osiemdziesiątych, kiedy to w Illinois w laboratorium CERL (ang. *U. S. Army Corps of Engineers' Construction Engineering Research Laboratory*) zapoczątkowano rozwój systemu do analiz przestrzennych na danych rastrowych [3]. Jak wiadomo, istniały wtedy dwa ścierające się wzajemnie podejścia do GIS. Jedno ceniło wyżej GIS zorientowany na obsługę danych wektorowych, a drugie - danych rastrowych. Nazwa GRASS (ang. *Geographic Resources Analysis Support System*, akronim nazwy oznacza trawę) jest swoistą kontynuacją nazewnictwa roślinnego, za powstałym kilka lat wcześniej systemem MOSS (ang. *Map Overlay and Statistical System*), którego akronim oznacza z kolei mech. MOSS był systemem zorientowanym na obsługę danych wektorowych.

Były to czasy olbrzymich, drogich i mało wydajnych komputerów oraz równie drogiego oprogramowania. W tym czasie dostępny był na rynku system GIS sprzedawany przez młodą firmę Environmental Systems Research Institute (ESRI), założoną przez Jacka Dangermonda, absolwenta Harvardu. System ten, ARCINFO, wzorowany na rozwijanych na Harvardzie SYMAP i ODYSSEY, wymagał jednak o wiele lepszego sprzętu, niż ten, którym dysponowało laboratorium CERL. Poza tym nie był tani.

Dlatego po przeanalizowaniu ówczesnego rynku i możliwości finansowych, powołany do tego celu zespół w CERL postanowił zarobić na wymarzony system GIS. Zaproponował swoim klientom drobne autorskie rozwiązania, przeniesione następnie do systemu UNIX. Rozwiązania GIS opracowywane przez laboratorium CERL zdobywały popularność i coraz większe uznanie wśród użytkowników systemu UNIX na wyższych uczelniach i w administracji publicznej. Równocześnie weszły na rynek małe komputery z tym systemem. Postanowiono wtedy, dla łatwiejszej instalacji, zebrać około 20 stworzonych dotąd programów w jeden pakiet. Nazwano go „GRASS”. Wkrótce okazało się, że zakup upragnionego systemu GIS nie jest już uzasadniony. Zamiast wykorzystać dostępne na rynku oprogramowanie, skupiono się na rozwoju własnego, rozwijanego na licencji Public Domain. Z biegiem czasu ESRI przestało wydawać wersję ARCINFO dla systemu UNIX i skupiło się na systemach prywatnych [3]. Oprogramowanie tej firmy różniło się też wtedy od systemu GRASS tym, że bazowało na obsłudze danych wektorowych i nie przetwarzało danych rastrowych.

GRASS jest wyjątkowym programem, któremu oczekiwania użytkowników nadały początkowy kształt i oczekiwania użytkowników kształtują jego przyszłość. W laboratorium CERL zdawano sobie sprawę, że skromne fundusze nie są w stanie zapewnić właściwego rozwoju systemu, dlatego postanowiono przyjąć już kiełkującą wśród użytkowników systemu UNIX filozofię open source i umożliwić każdemu tworzenie nowych modułów systemu GRASS.

Społeczność skupiona wokół systemu GRASS okazała się być silną i aktywną. Ewoluowała w kilka wpływowych organizacji, mających znaczenie także dla dzisiejszego kierunku rozwoju systemów informacji geograficznej. Na początku była to fundacja OGF (ang. *Open GRASS Foundation*, 1992r.), po dwóch latach, aby podkreślić swe szersze zainteresowania, organizacja ta zmieniła nazwę na Open GIS Foundation [2]. Wynikało to z faktu, iż zajmowała się ona również kwestią interoperacyjności danych oraz tworzeniem specyfikacji dostępu do różnorodnych baz danych OGIS (ang. *Open Geodata Interoperability Specification*). Konieczność dalszej pracy nad interoperacyjnością i wymianą danych pomiędzy systemami GIS oraz ograniczenia prawne organizacji o statusie fundacji doprowadziły do przekształcenia fundacji w konsorcjum OGC (ang. *Open Geospatial Consortium, Inc*). Jest to organizacja skupiająca szereg firm i instytucji rządowych z całego świata oraz osoby prywatne, której zadaniem jest kształtowanie obecnych standardów wymiany danych.

Zadaniem Open GRASS Foundation było przejęcie opieki nad rozwojem projektu. Zastąpiła ona powstały w 1986r. pierwszy komitet sterujący projektu GISC (ang. G*RASS Interagency Steering Committee*). Powodem było między innymi niezadowolenie amerykańskiego komercyjnego rynku GIS z faktu wspierania przez Rząd Federalny wolnego oprogramowania, uważanego za konkurencję. Konsekwencją kolejnych przemian powołanej wtedy fundacji, był brak w tym okresie oficjalnych wydań systemu GRASS.

Szczęśliwie, w 1997r. GRASS, zadomowił się na Uniwersytecie Baylor w Teksasie. Był on już wtedy dostępny dla systemów GNU/Linux. Krótko potem przeniósł się na Uniwersytet w Hanowerze i do dziś jest rozwijany przez międzynarodowy zespół programistów (ang. *GRASS Development Team*), pod pieczą Markusa Netelera. W 1999r. zmieniono także licencję programu z public domain na GNU GPL. Wraz ze swoim opiekunem GRASS podróżuje po świecie, a obecnie jego domem jest Fundacja Edmunda Mach'a w Trydencie [5].

Projekt GRASS jest również jednym z założycieli znanej na całym świecie fundacji OSGeo (ang. *Open Source Geospatial Foundation*), wspierającej rozwój wolnego oprogramowania o otwartym kodzie źródłowym dla zastosowań geoprzestrzennych. Powstała w 2006 roku fundacja [6], dostarcza między innymi infrastrukturę techniczną dla przechowywania repozytoriów kodu źródłowego projektów, prowadzenia list mailingowych [7] i forów użytkowników. Jej serwery znajdują się w Stanach Zjednoczonych. Z biegiem czasu w kolejnych krajach powstają oddziały fundacji wspierające rozwój lokalnych społeczności, są także oddziały OSGeo nie związane z położeniem geograficznym, np. Women Chapter [8].

Raz w roku, zawsze na innym kontynencie, odbywa się międzynarodowa konferencja OSGeo. Spotykają się na niej programiści, a ostatnio również i zwykli użytkownicy wolnego oprogramowania, by wymienić się doświadczeniami z różnych stron świata. Ostatnia konferencja miała miejsce w Barcelonie we wrześniu 2010 i przyciągnęła 869 uczestników.

Projekt GRASS był na niej szeroko reprezentowany na:

- warsztatach W-14 „Practical introduction to GRASS” [9],
- dziewięciu posterach [10a,10b],
- oraz licznych prezentacjach [11a,11b].

Jedna z nich dotyczyła porównania możliwości modułów systemu GRASS i innego oprogramowania służącego filtracji danych lidarowych [12]. Świadczy to nie tylko o dużym zainteresowaniu projektem, ale też o znacznym zaangażowaniu społeczności systemu GRASS w jego popularyzację.

2.3. PSC (Project Steering Commitee)

Rok 2006 był przełomowym w historii projektu GRASS. Na potrzeby OSGeo został powołany dziesięcioosobowy Komitet Sterujący Projektu (ang. *GRASS Project Steering Commitee – PSC*) [13], na którego czele, na drodze jawnego głosowania, stanął Markus Neteler [14]. Jednym z członków PSC jest Polak, Maciej Sieczka. Oprócz dbania o ogólny rozwój projektu, głównym zadaniem PSC jest przyznawanie programistom praw do modyfikacji kodu źródłowego systemu GRASS.

2.4. Strona internetowa projektu GRASS

W 2008 roku strony projektu GRASS zostały przeniesione na serwery fundacji OSGeo [15]. Społeczność, dla zapewnienia sobie płynnego dostępu do stron systemu GRASS z różnych części świata, utrzymuje serwery lustrzane na pięciu kontynentach [16]. Dwa z nich znajdują się w Polsce; jeden na Uniwersytecie Wrocławskim [17], a drugi w Krakowie [18]. Mirrory są codziennie synchronizowane ze stroną główną. Utrzymywanie dokładnych kopii stron internetowych systemu GRASS odciąża główny serwer i zapewnia nieprzerwany dostęp użytkownikom do plików źródłowych, instalacyjnych oraz dokumentacji.

2.5. Wiki

Oprócz oficjalnej strony internetowej projektu, będącej jego wizytówką, istnieją również strony pisane przez użytkowników dla nich samych, zbudowane na zasadzie wiki[1]. Do umieszczania informacji na tych stronach ma prawo każdy, po założeniu konta i zalogowaniu się. Zmiany tu wprowadzone widoczne są od razu i nie podlegają wstępnej weryfikacji. Nad kontrolą treści zamieszczanych na wiki pełni pieczę troje administratorów, którzy w razie potrzeby blokują spamerów i cofają wprowadzone zmiany. Na stronach założony jest system RSS[2], który pozwala na bieżąco śledzić wszelkie ruchy. Każda strona wiki ma zakładkę z historią, z której można się dowiedzieć, kto i kiedy zamieścił na niej konkretną informację.

Przyjęto zasadę, że strony GRASS-Wiki prowadzone są w języku angielskim i mogą być tłumaczone na inne języki, natomiast całkiem nowe treści nie powstają na tych stronach w innych językach. Do tego celu służą inne, zakładane przez użytkowników strony, np.:

- `http://grass.fsv.cvut.cz/wiki`,
- `http://www.wgug.org`,
- `http://www.grass-gis.pl`.

Strony wiki w niewielkim procencie przetłumaczone są na język polski i tylko od potrzeb polskich użytkowników zależą postępy ich tłumaczenia. Dla łatwiejszego poruszania się, strony oznaczane są kategoriami; wszystkie przetłumaczone dotąd na język polski są opatrzone kategorią „Polski”.

Użytkownicy systemu GRASS zamieszczają na stronach wiki informacje, które ich zdaniem są istotne dla społeczności. Na przykład, użytkownicy piszący dodatki do systemu GRASS (ang. *add-ons*) zamieszczają o nich krótką informację [19]. W ten sposób na bieżąco aktualizowana jest informacja o modułach użytkowników, dodawanych do repozytorium add-ons projektu. Jeśli dany moduł

1 Wiki - rodzaj stron internetowych, które można z łatwością tworzyć, edytować czy modyfikować za pomocą przeglądarki internetowej.

2 RSS - czytnik kanałów (ang. feed reader lub news aggregator), czyli program komputerowy do czytania kanałów informacyjnych w formatach RSS lub Atom, opartych na języku XML. W praktyce umożliwia internautom pobieranie i śledzenie zmian wprowadzanych na wyposażonych weń stronach internetowych.

zostanie dołączony do kodu głównego systemu GRASS, informacja taka również zostaje umieszczona na tej stronie.

2.6. Komunikacja

Użytkownicy systemu GRASS w dalszym ciągu komunikują się ze sobą za pomocą najstarszego czatu[3]: IRC (kanał #grass na serwerze freenode: `irc://irc.freenode.net/grass`), a także list dyskusyjnych [20] i forów internetowych. GRASS, jako aktywnie rozwijający się projekt, posiada wiele tematycznych list dyskusyjnych; oddzielną dla programistów, użytkowników, tłumaczy etc. Można na nich prosić o poradę, a osoby, które jej udzielają, robią to na zasadzie wolontariatu. Dlatego tak ważny jest podział tematyczny list, jak również zachowanie się na nich nowych użytkowników. Właściwym postępowaniem jest aby przed wysłaniem pytania zerknąć do historii listy, czy dany problem nie był ostatnio poruszany i czy nie znajdziemy na niego odpowiedzi bez angażowania uwagi setek użytkowników. Można to zrobić korzystając z serwera Nabble [21], na którym przegląda się listy mailingowe na zasadzie forum dyskusyjnego. Jest to również o wiele szybszy sposób na uzyskanie potrzebnej informacji.

Rok temu powstało także polskie forum GRASS [22]. Więcej informacji na temat sposobów komunikacji można znaleźć na stronie projektu [23].

2.7. Nieskrępowany dostęp

Użytkownicy systemu GRASS mają, zgodnie z licencją, dostęp do jego kodu źródłowego we wszystkich niemalże wersjach, od tych historycznych, poprzez obecnie wydane 6.4., do wersji rozwojowych 6.5 i 7.0, będących w przygotowaniu i z tego względu nie zalecanych do pracy produkcyjnej. Jednak użytkownicy często sięgają po wersje rozwojowe, ponieważ dostarczają one nowe narzędzia i usprawnienia, które nie zdążą już zostać włączone do wydania stabilnego. Przykłady: modeler graficzny [24], rozwijany dla GRASS 6.5 i 7.0 przez Martina Landę, służący do projektowania powtarzalnych procesów przetwarzania i analizy danych przestrzennych, moduł interpolacji metodą krigingu, grupa nowych modułów do obróbki zobrazowań satelitarnych. Pełna lista nowości w GRASS 7 jest dostepna pod adresem: `http://trac.osgeo.org/grass/wiki/Grass7/NewFeatures`.

3 Czat (ang. chat – rozmowa) - jedna z usług internetowych, umożliwiająca komunikację na żywo za pomocą słowa pisanego.

2.8. Repozytorium kodu źródłowego

Kod źródłowy systemu GRASS można przeglądać bezpośrednio w Internecie [25], za pomocą przeglądarki Trac [26], lub pobrać jego kopię na swój dysk lokalny.

Jak w każdym szanującym się projekcie programistycznym, do sprawnego zarządzania kodem źródłowym programu stosuje się system kontroli wersji. Do tego celu istnieje kilka narzędzi, takich jak CVS, SVN, czy GIT. Do obsługi kodu źródłowego GRASS używany jest SVN (ang. *Subversion*) [27]. Pozwala on na zarządzanie wersjami programu, wprowadzanie zmian wielu programistom jednocześnie, pozwala też na odtwarzanie poprzednich wersji, jeśli zmiany wprowadzono przez pomyłkę lub powodują one błędne działanie systemu.

Przykładowym klientem SVN dla systemu Windows jest TortoiseSVN, dostępny oczywiście również na wolnej licencji GPL. Po zainstalowaniu, Tortoise integruje się z przeglądarką plików a jego funkcje dostępne są bezpośrednio w przeglądarce z menu kontekstowego (pod prawym klawiszem myszy).

W odróżnieniu od większości programów GIS, GRASS nie jest monolityczną aplikacją lecz zbiorem oddzielnych, wyspecjalizowanych, ale współpracujących ze sobą i "mówiących jednym językiem" modułów. Taka architektura pozytywnie wpływa na stabilność środowiska pracy GRASS, ewentualny krytyczny błąd w jednym module nie ma żadnego bezpośredniego wpływu na samo środowisko ani na pozostałe moduły.

Dostępność źródeł programu zapewnia społeczności GRASS możliwość korzystania z danej funkcji programu bezpośrednio po jej dodaniu przez któregokolwiek z programistów oraz możliwość testowania własnych modyfikacji programu, zanim zostaną dodane do repozytorium. Korzystanie z tych możliwości wymaga jednak samodzielnego skompilowania kodu źródłowego do pakietów binarnych, co w systemach GNU/Linux nie nastręcza trudności, jest natomiast nieco bardziej skomplikowaną operacją w systemach Windows. Dlatego z pomocą przychodzi znowu społeczność użytkowników, tym razem czeska społeczność, która udostępnia codziennie kompilowane pakiety binarne dla tych systemów na serwerach Czeskiego Uniwersytetu Technicznego w Pradze. Pakiety te kompilowane są zarówno dla GRASS 6.4 [28] jak i 6.5 [29] oraz 7.0, co ułatwia innym użytkownikom systemu GRASS na systemie Windows testowanie nowych funkcji programu i zgłaszanie uwag czy błędów na bieżąco.

Repozytorium zostało podzielone na części:

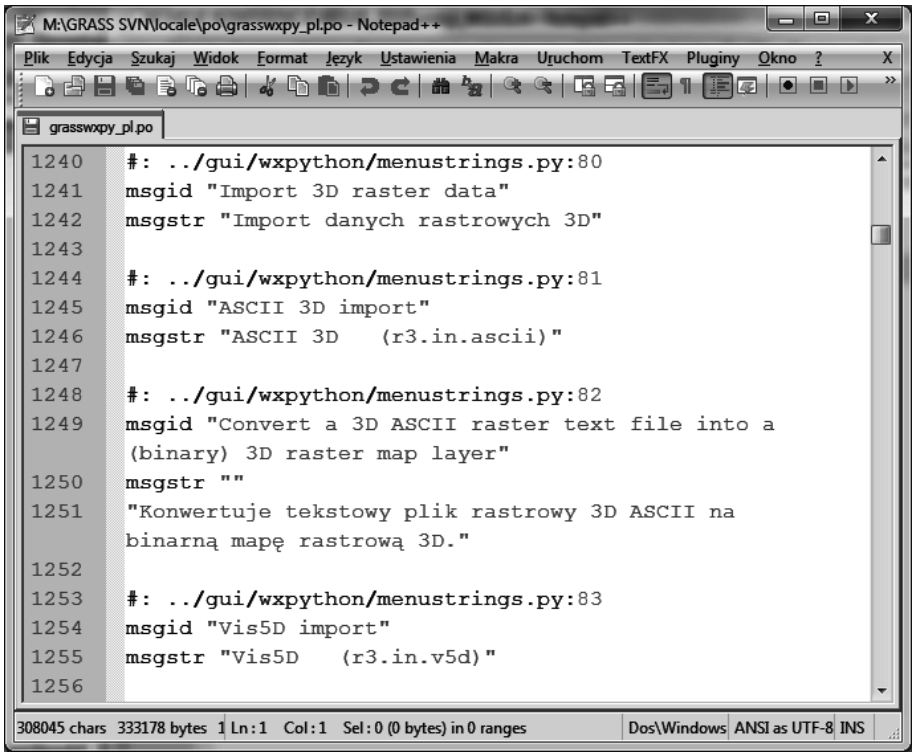
- GRASS `http://svn.osgeo.org/grass/grass/trunk/` - zawierającą kod źródłowy systemu.
- Dodatki (AddOns) `http://svn.osgeo.org/grass/grass-addons/` - w której znajdziemy moduły napisane przez użytkowników, nie dołączone jeszcze do oficjalnej wersji programu.
- Strona internetowa `http://svn.osgeo.org/grass/grass-web/`.
- W tym roku wydzielono także czwartą część repozytorium, mianowicie: Materiały promocyjne `http://svn.osgeo.org/grass/grass-promo/` - ze względu na przechowywanie tam plików o dużych rozmiarach, np. posterów, których aktualizacja pociągała za sobą wcześniej konieczność aktualizacji kopii SVN u wszystkich programistów i ściągania dużej ilości danych nie mających związku z kodem źródłowym.

2.9. Życzenia użytkowników

Do zgłaszania życzeń i błędów znalezionych w programie wykorzystywany jest system śledzenia zgłoszeń Trac [30], który umożliwia zakładanie, przeglądanie, przydzielanie programistom i zamykanie tych zgłoszeń po rozwiązaniu zadania. Jednocześnie pozwala to przeglądać ich stan i historię. Zgłoszenia reprezentowane są przez tzw. bilety, przypisywane konkretnemu programiście, który zajmuje się danym zagadnieniem w projekcie. W formularzu biletu wypełnia się również odpowiednie pola precyzujące, której części kodu on dotyczy, jakiego systemu operacyjnego, której wersji programu, etc.

Przed wypisaniem biletu należy przeszukać historię zgłoszeń, czy dany problem nie był wcześniej poruszony, aby nie dublować biletów. Wszelka aktywność w systemie biletowania wysyłana jest na listę dyskusyjną GRASS-trac i rozsyłana emailem do wszystkich użytkowników ją subskrybujących oraz pozostaje w systemie Trac. Trafia również do kanału IRC.

Wszystko to po to, aby można było śledzić postępy i dyskusje towarzyszące rozwiązywaniu danego problemu. Między innymi w ten właśnie sposób społeczność wolnego oprogramowania zdobywa wiedzę i nieprzerwanie się uczy. Aby móc zgłaszać błędy w tym systemie, użytkownik musi posiadać login osgeo i hasło. Zapewnia to wystarczającą weryfikację użytkowników i podstawową ochronę przed spamerami, na działanie których poważny projekt nie może sobie przecież pozwolić. Użycie jednego loginu, pozwalającego na zgłaszanie błędów we wszystkich projektach będących pod parasolem fundacji OSGeo, oszczędza również cenny czas użytkowników i zachęca do dbania o wiele projektów. Inną drogą zgłaszania życzeń i błędów jest lista dyskusyjna GRASS-dev [31], jednak tam zgłoszone mogą zostać przypadkiem nie zauważone pośród setek innych wiadomości. Zgłoszenie w systemie Trac daje większą gwarancję, że opisany problem znajdzie rozwiązanie.



Rys. 2.1 Plik tłumaczenia w programie Notepad++

2.10. Polonizacja

Polonizacja systemu GRASS nie należy do łatwych zadań. Nie jest to typowy system GIS, a zestaw (około 400) wyspecjalizowanych narzędzi do analiz przestrzennych, dlatego wymaga od tłumacza znajomości szerokiego wachlarza specjalistycznej nomenklatury z wielu dziedzin nauki. Tłumaczenie GRASS obejmuje trzy obszary: tłumaczenie GUI[4], tłumaczenie komunikatów poleceń, tłumaczenie dokumentacji. Obecnie prowadzone są prace nad wszystkimi trzema obszarami.

Pracę utrudnia brak narzędzia, które umożliwiałoby tłumaczenie kontekstowe, tak ważne nie tylko z powodu złożoności zagadnień technicznych, ale również ze względu na bogatą gramatykę języka polskiego. Innymi słowy, przy tak skomplikowanym programie nie sposób tłumaczyć go poprawnie, nie widząc jednocześnie budowy konkretnego okna GUI.

Pliki do tłumaczeń komunikatów modułów oraz GUI jest po cztery dla każdego języka. Znajdują się one w źródłach w katalogu /locale/po:

- grasslibs_LANG.po - zawiera ciągi tekstowe z bibliotek
- grassmods_LANG.po -zawiera ciągi tekstowe z modułów
- grasstcl_LANG.po - zawiera ciągi tekstowe z GUI Tcl/Tk
- grasswxpy_LANG.po - zawiera ciągi tekstowe z GUI wxgrass/wxPython,

"LANG" oznacza dwuliterowy skrót języka, np. grasswxpy_pl.po dla języka polskiego.

Pliki odpowiedzialne za język polski przetłumaczone są odpowiednio w 64%, 37%, 80% i 90% (stan na dzień: 1.09.2010). Jednak ciągle wymagają one sprawdzenia poprawności merytorycznej oraz gramatycznej.

W przypadku braku narzędzia do kontekstowego tłumaczenia GUI, najlepszym wydaje się być ulubiony edytor tekstowy (rys. 2.1) i jednocześnie otwarty GRASS, aby można było kontrolować poprawność tłumaczenia.

4 GUI (ang. Graphical User Interface) - Graficzny Interfejs Użytkownika.

Oprócz tego, istnieją również programy do tłumaczeń na licencjach zapewniających wolność użytkowania i otwartość kodu, którymi można się posłużyć. W przeciwieństwie do edytora tekstowego, można za ich pomocą generować bazę tłumaczeń z plików już znajdujących się na dysku, a następnie skorzystać z podpowiedzi tłumaczenia, czyli poleceń czy ich instrukcji, które już raz zostały przetłumaczone, a są identyczne lub podobne do tych nieprzetłumaczonych. Stanowi to dużą pomoc w przypadku tłumaczenia łańcuchów tekstowych GUI, pozwalając użyć ponownie wcześniej przetłumaczone polecenia czy opisy starszego GUI Tcl/Tk w GUI bazującym na wxPythonie.

Programami tego typu są:

- poEdit (rys. 2.2) dla systemów Windows,
- Lokalize (dawny KBabel) dla środowiska KDE SC, czy
- wieloplatformowy Virtaal (rys. 2.3).

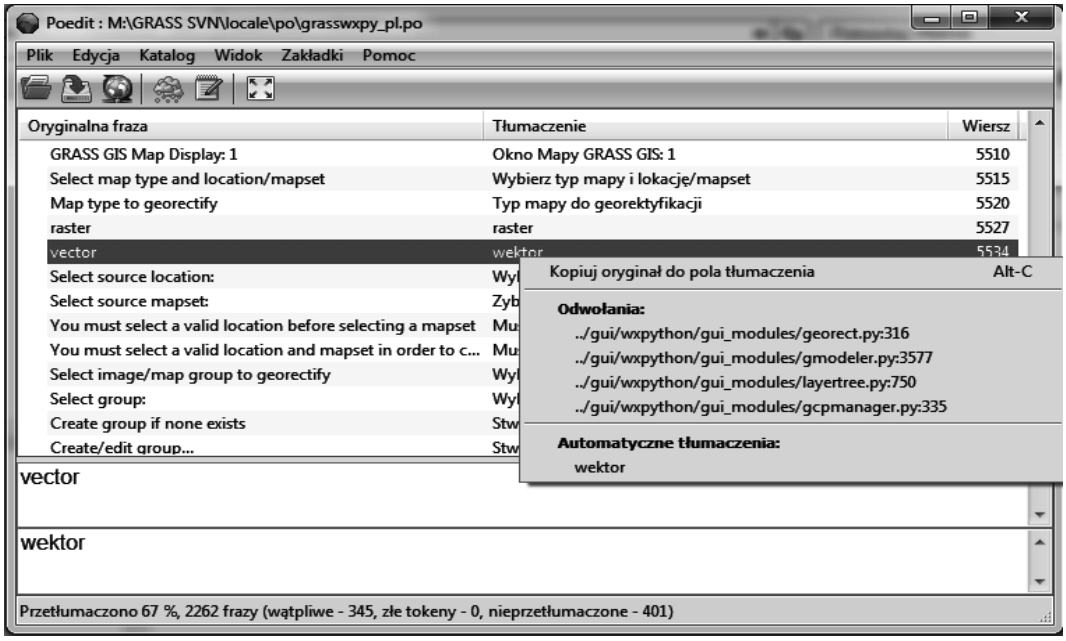
Wszystkie są programami wolnymi o otwartym kodzie źródłowym. PoEdit wydany jest na licencji MIT [32], Lokalize oraz Virtaal opatrzono licencją GNU General Public License.

Warto też wspomnieć o możliwości tłumaczenia online, czyli bezpośrednio w Internecie. Po ubiegłorocznych dyskusjach na liście GRASS-translations, wybrano do tego celu portal Transifex (rys. 2.4) [33]. Można tu, bez instalowania jakiegokolwiek oprogramowania, tłumaczyć wszystkie cztery pliki *.po lub pobrać je do tłumaczenia na swój dysk, jednocześnie blokując je przed edycją innych użytkowników aż do skończenia pracy.

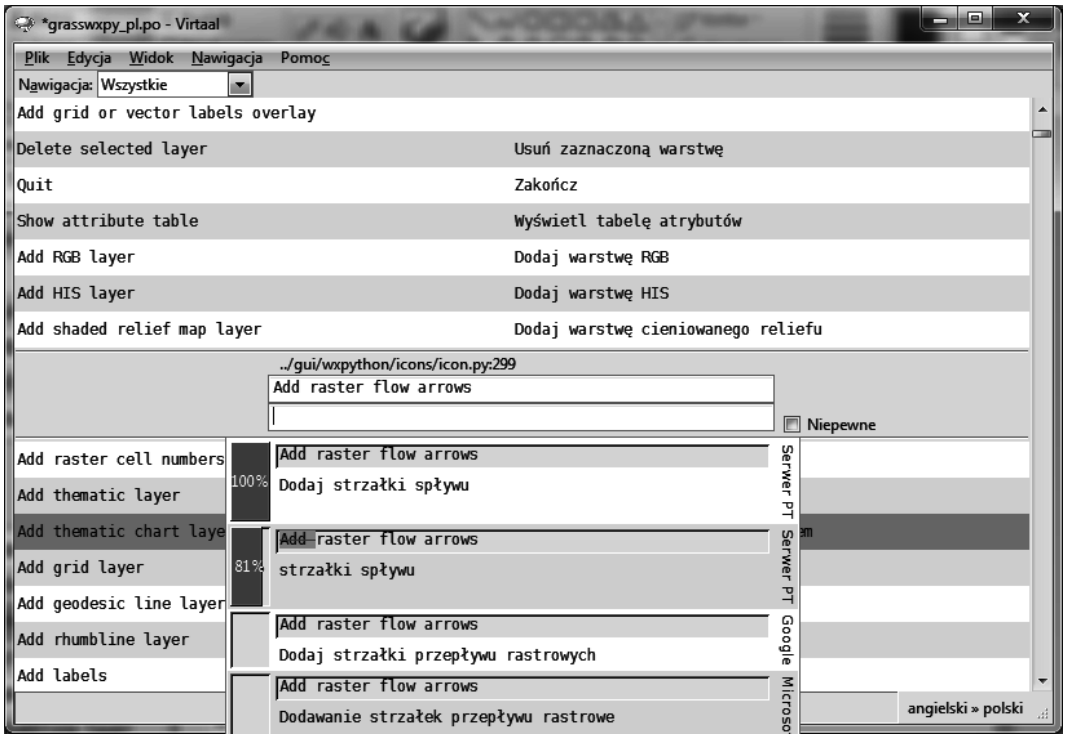
Narzędzia do tłumaczeń online mają, siłą rzeczy, mniej możliwości. Na przykład zwykle nie posiadają bazy tłumaczeń i nie generują podpowiedzi. Zaletą jest to, że można ich użyć bezpośrednio, na aktualnych danych, z każdego komputera podłączonego do sieci Internet.

Przy tłumaczeniach, jeśli nie korzysta się z portalu Transifex, ważną zasadą jest ogłoszenie na liście dyskusyjnej GRASS-translations [34] informacji o rozpoczęciu tłumaczenia. Chodzi o to, aby dwie osoby nie pracowały jednocześnie nad tym samym plikiem.

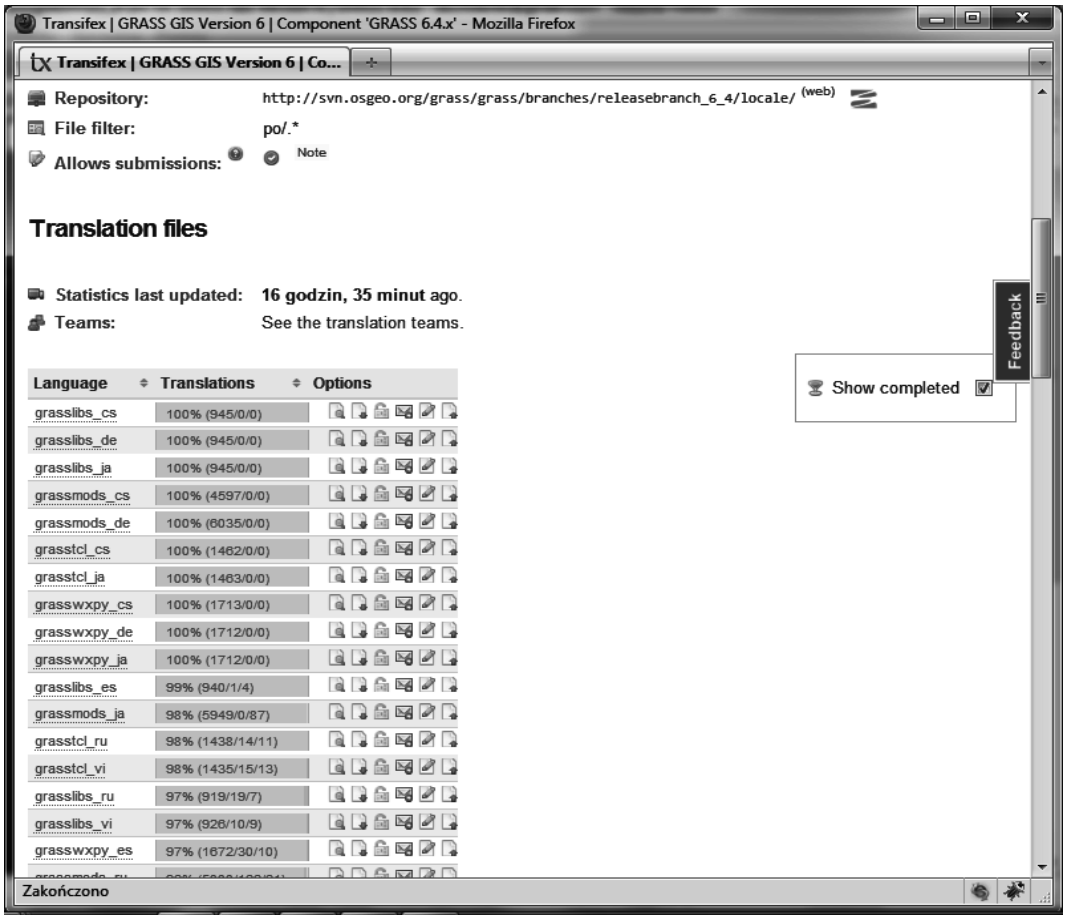
Ponieważ GRASS jest żywym projektem, tłumaczone pliki należy przesyłać do repozytorium często, aby uniknąć konfliktów z plikami źródłowymi. Można to zrobić wysyłając zmodyfikowany plik e-mailem do dowolnego programisty GRASS.



Rys. 2.2 poEdit



Rys. 2.3 Virtaal



Rys. 2.4 Portal Transifex

Ważną częścią spolszczania programu jest tłumaczenie podręcznika użytkownika. GRASS jest jednym z niewielu programów, które posiadają pomoc do każdej niemal funkcji. Są to spójne i jednorodne pliki HTML opisujące każdy z modułów, dla wygody użytkowników tworzone według jednego schematu.

W plikach pomocy znajdziemy najczęściej następujące sekcje:

- Nazwa (ang. *Name*)
- Słowa kluczowe (ang. *Keywords*)
- Składnia (ang. *Synopsis*)
- Opcje (ang. *Flags*)
- Parametry (ang. *Parametres*)
- Opis (ang. *Description*)
- Definicje (ang. *Definitions*)
- Uwagi (ang. *Notes*)
- Przykłady (ang. *Examples*)
- Zobacz także (ang. *See also*)
- Autor (ang. *Author*)

Zastosowanie schematu do opisu każdego z modułów pomaga użytkownikom w szybki sposób wyszukać potrzebne informacje na temat działania danej funkcji systemu GRASS.

Pliki pomocy dostępne są zarówno w Internecie, np. na stronie głównej projektu [35], jak i zainstalowane razem z programem, otwierane lokalnie w przeglądarce internetowej (menu Pomoc - cały podręcznik), oraz bezpośrednio z okna każdego modułu (karta dla tego modułu). Przy czym jest to pomoc w języku angielskim, częściowo przetłumaczona na języki czeski oraz włoski.

W języku polskim tłumaczenie pomocy wybranych modułów znajdziemy w skrypcie “Podstawy systemu GRASS” [Podstawy systemu GRASS - skrypt w formacie *.pdf, ok. 750k, język polski. Autor: Paweł Netzel], jednak żadne przetłumaczone karty pomocy nie zostały jeszcze włączone do kodu źródłowego systemu GRASS, a techniczna obsługa tłumaczonych plików pomocy pozostaje do tej pory sprawą otwartą.

2.11. Podsumowanie

Projekt GRASS jest rozwijany przez międzynarodową społeczność programistów, tłumaczy, osoby piszące dokumentację, przygotowujące instalatory na różne systemy operacyjne. Osoby zaangażowane w projekt wykonują swoją pracę na zasadzie wolontariatu, zazwyczaj pracują nad tym, co im samym jest w danej chwili potrzebne do lepszego wykorzystania systemu GRASS w ich codziennych służbowych obowiązkach. Robią to chętnie dlatego, że te drobne udoskonalenia są potrzebne także innym osobom, które z kolei pracują nad innymi cząstkami projektu. Zdarza się, że firmy, organizacje czy jednostki administracji publicznej używające systemu GRASS w codziennej pracy, potrafią wygospodarować środki finansowe na napisanie konkretnej, potrzebnej im funkcji lub udoskonalenie już istniejącej. Wtedy system ma szansę rozwijać się szybciej a programiści mają dodatkową, finansową motywację do pracy. Stosunkowo rzadko zdarza się, że w ten sposób motywowani są tłumacze czy osoby piszące dokumentację oraz pakujące program dla użytkowników. Nic nie stoi wszakże na przeszkodzie, aby pójść za przykładem programu Quantum GIS, gdzie kompletne tłumaczenie podręcznika i GUI programu na język niemiecki zostało zamówione i opłacone przez władze Szwajcarskiego Kantonu Solury. Wyniki tej pracy są dostępne dla każdego użytkownika programu, który potrzebuje przetłumaczonych materiałów. Jak można zauważyć, kierunek i prędkość rozwoju projektów opartych na wolnych licencjach zależy wyłącznie od woli użytkowników. Dlatego programy oparte na tej i podobnych licencjach rozwijają się szybciej niż oprogramowanie prywatne, należące do poszczególnych firm. Co więcej, projekty te nie mogą zbankrutować ani utracić płynności finansowej, a przestaną się rozwijać tylko wtedy, jeśli utracą swego ostatniego użytkownika.

Literatura

[1] Neteler M., Mitasova H., 2008, *Open source GIS: A GRASS GIS approach. Third Edition*, The International Series in Engineering and Computer Sciences: Volume 773. Springer New York Inc., ss 406
[2] Schell D., 1993, *The Open GIS Foundation*. GRASSClippings – The Journal of Open Georaphic Information Systems, 7(2), s. 4–5.
[3] Westervelt J., *GRASS Roots*. Proceedings of the FOSS/GRASS Users Conference - Bangkok, Thailand, 12-14 September 2004, ss. 10

Źródła internetowe:

[4] https://svn.osgeo.org/grass/grass/branches/releasebranch_6_4/COPYING
[5] http://gis.fem-environment.eu/
[6] http://www.osgeo.org
[7] http://lists.osgeo.org/mailman/listinfo
[8] http://wiki.osgeo.org/wiki/Women_Chapter
[9] http://2010.foss4g.org/workshop.php
[10a] http://2010.foss4g.org/presentations_pos_sel.php,
[10b] http://2010.foss4g.org/presentations_aca_pos.php
[11a] http://2010.foss4g.org/presentations_gen_sel.php,
[11b] http://2010.foss4g.org/presentations_aca_ora.php
[12] http://2010.foss4g.org/presentations_show.php?id=3248
[13] http://grass.osgeo.org/wiki/GRASS_Project_Steering_Commitee
[14] http://lists.osgeo.org/pipermail/grass-psc/2006-December/000143.html
[15] http://grass.osgeo.org
[16] http://grass.osgeo.org/mirrors.php
[17] http://grass.meteo.uni.wroc.pl/index.php
[18] http://grass.wodgik.malopolska.pl/index.php
[19] http://grass.osgeo.org/wiki/GRASS_AddOns/
[20] http://lists.osgeo.org/mailman/listinfo
[21] http://osgeo-org.1803224.n2.nabble.com/GRASS-f1837859.html
[22] http://forum.grass-gis.pl/
[23] http://grass.osgeo.org/community/support.php
[24] http://grass.osgeo.org/wiki/WxGUI_Modeler/pl
[25] http://svn.osgeo.org/grass/grass/trunk/
[26] http://trac.osgeo.org/grass/browser/grass
[27] http://subversion.tigris.org/
[28] http://josef.fsv.cvut.cz/wingrass/grass64/
[29] http://josef.fsv.cvut.cz/wingrass/grass65/
[30] http://trac.osgeo.org/grass/report
[31] http://lists.osgeo.org/mailman/listinfo/grass-dev
[32] http://www.opensource.org/licenses/mit-license.html
[33] http://www.transifex.net/projects/p/grass6/c/grass64/
[34] http://www.osgeo.org/mailman/listinfo/grass-translations
[35] http://grass.osgeo.org/grass64/manuals/html64_user/index.html

3. Interfejsy graficzne w systemie GRASS

Robert Szczepanek

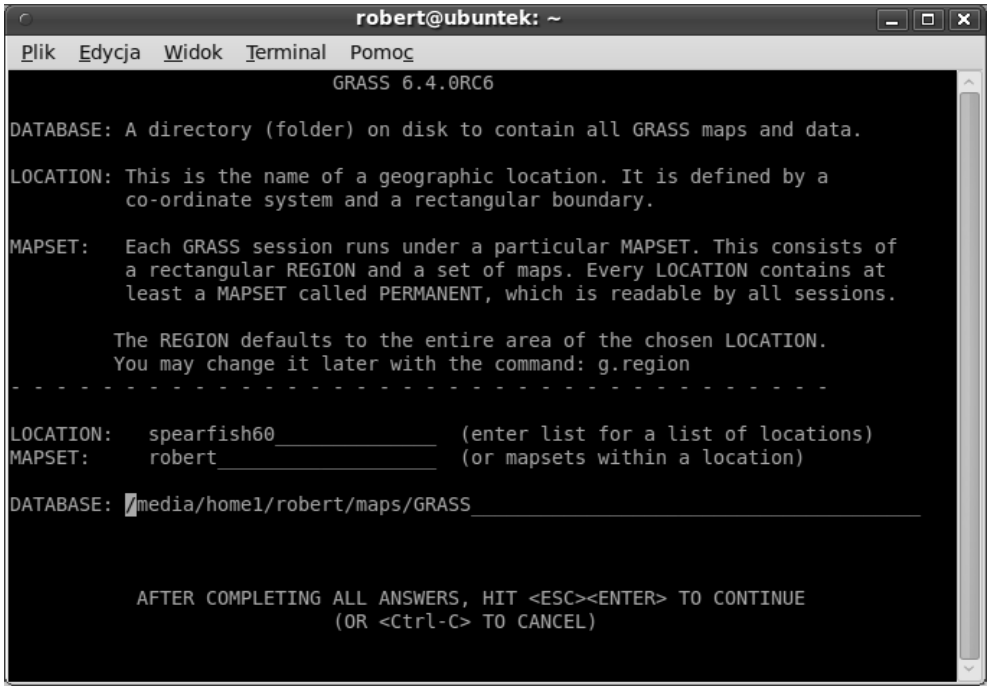
3.1. Wstęp

We współczesnym świecie jedynie rozwiązania ergonomiczne mają szanse na przetrwanie w walce z coraz bardziej agresywną konkurencją. Użytkownicy zwracają uwagę już nie tylko na możliwości urządzeń, ale coraz częściej również na ich estetykę i wygodę użytkowania. Dotyczy to również systemów informatycznych. Wydaje się, że bezpowrotnie minęły czasy, gdy obsługa komputera była domeną wąskiego grona specjalistów. Komputery coraz częściej są postrzegane jako nieodłączny element życia codziennego. Wymusza to na ich twórcach nadążanie za ciągle rosnącymi wymaganiami użytkowników.

Czy system GRASS, będący jednym z bardziej znanych systemów informacji przestrzennej, ale jednocześnie mający już swoje lata, nadąża za tymi trendami? I czy na pewno musi za nimi nadążać? Czy zawsze to co ładne i podane w przystępnej wizualnie formie jest najbardziej ergonomiczne?

3.2. Interfejs tekstowy

Początkowo komunikacja z komputerami realizowana była wyłącznie poprzez linie komend. W tych właśnie zamierzchłych czasach narodził się GRASS (ang. *Geographic Resources Analysis Support System*). Jego pierwsi użytkownicy nie mieli wielkiego wyboru, gdyż graficzne interfejsy użytkownika (ang. *Graphical User Interfaces*) były w początkowej fazie rozwoju. Dla ówczesnych użytkowników nie miało to prawdopodobnie większego znaczenia, gdyż w znakomitej większości użytkownikami byli sami programiści.



Rys. 3.1 Uruchomienie programu GRASS w trybie tekstowym.

Uruchomienie systemu GRASS wymagało znajomości podstawowych komend programu oraz lokalizacji plików z mapami. Najnowszą wersję programu GRASS również można uruchomić w takim trybie wydając w teminalu komendę:

```
grass64 -text
```

Użytkownikowi ukazywał się obraz pokazany na rysunku 3.1. Wpisując kolejno: bazę danych, lokację oraz mapset, uruchamiany był tryb interaktywny pracy z programem.

Po tych operacjach każda komenda, wraz z ewentualnymi ścieżkami dostępu, nazwami map i opcjami, była wpisywana ręcznie. Oczywiście należało wiedzieć jaką komendę należy wydać i jaką ma ona składnię. Patrząc z punktu widzenia współczesnego użytkownika, najczęściej wychowanego na systemie operacyjnym MS Windows, był to system skrajnie trudny do pracy.

Wydaje się zresztą, że stosunkowo mała popularność GRASS-a, biorąc pod uwagę jego możliwości, brała się w dużej mierze z faktu braku wersji pracującej pod kontrolą systemu operacyjnego Windows.

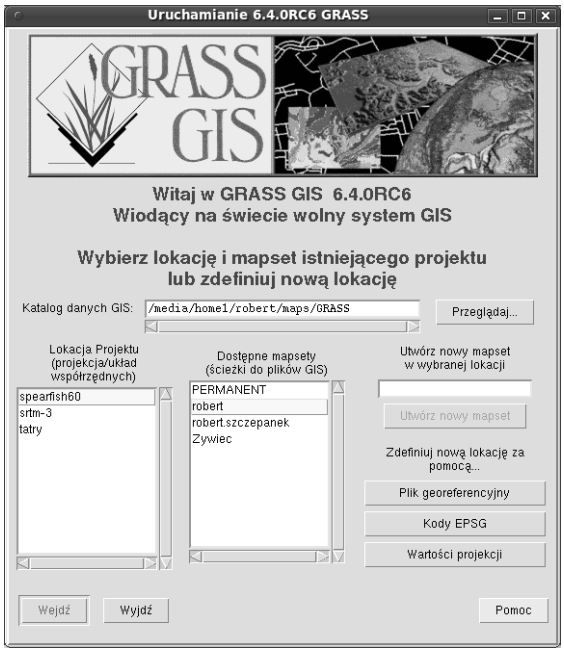
3.3. Interfejsy graficzne

W czasach, gdy głównym środowiskiem pracy programu GRASS był system operacyjny Unix, rozpoczęto prace nad graficznym interfejsem użytkownika (ang. *GUI*) opartym na wolnej, wieloplatformowej bibliotece Tcl/Tk. Do uruchomienia programu w tym trybie należy wydać komendę:

```
grass64 -tcltk
```

Po uruchomieniu programu w trybie graficznym, oczom użytkownika ukazywało się znacznie bardziej przyjazne środowisko pracy (rys. 3.2). Wygodniejsze i prostsze stało się wyszukiwanie map. Łatwiejszy stawał się początek pracy z programem.

Praca z programem realizowana jest poprzez kilka okien graficznych (rys. 3.3). Pierwsze z nich to tradycyjne okno terminala. Drugie okno, będące menadżerem warstw, umożliwia zarządzanie wyświetlanymi warstwami.



Rys. 3.2 Uruchomienie programu GRASS 6.4 w trybie tcltk.

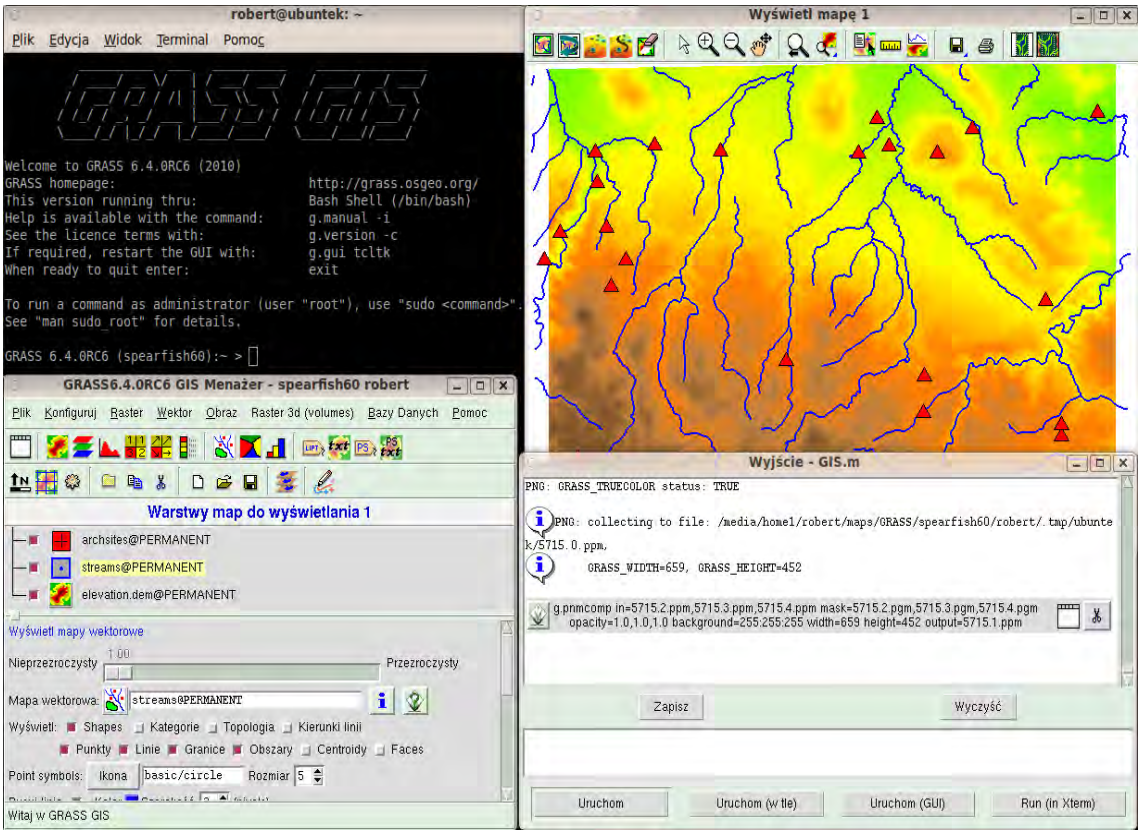
Można dodać lub usunąć dowolną warstwę lub też zmienić właściwości jej wyświetlania.

Ze względu na dużą liczbę opcji związanych np. z warstwami wektorowymi, okno to jest stosunkowo rozbudowane. Dodatkowo w oknie tym zostało umieszczone menu ułatwiające wyszukiwanie komend programu.

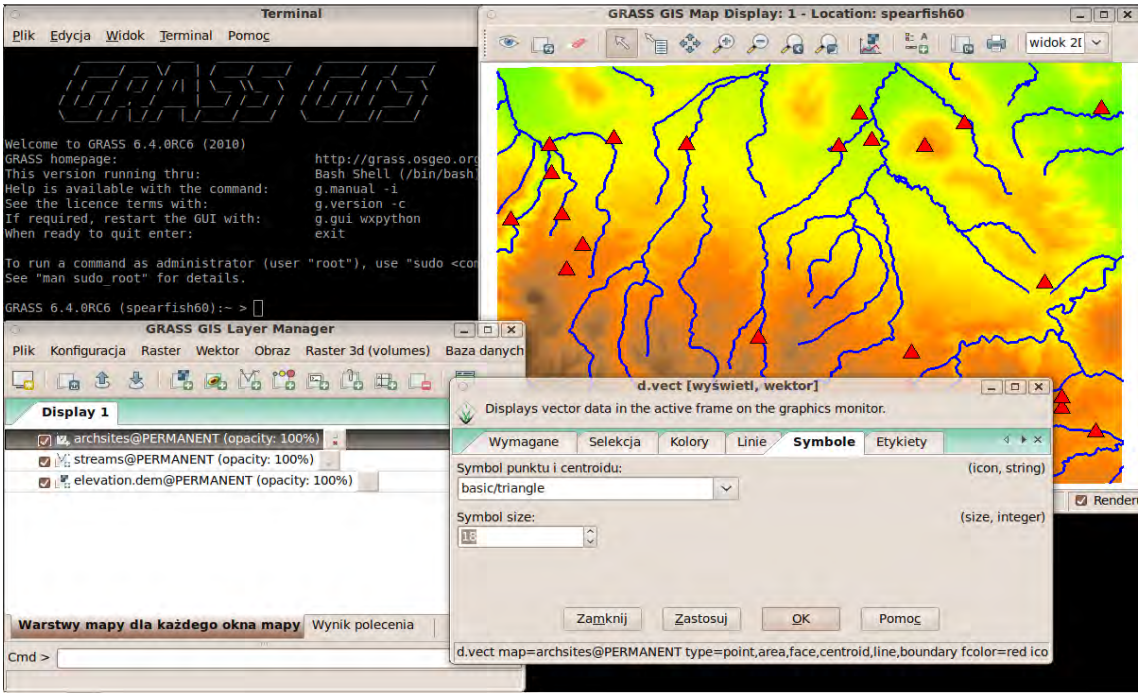
Trzecie z okien służy do wyświetlania map. GRASS umożliwia równoczesne wyświetlanie wielu takich okien (monitorów graficznych). W ostatnim z okien wyświetlane są komunikaty zwracane przez program. Dostęp do monitorów graficznych za pośrednictwem GUI umożliwił bardziej elastyczną pracę z mapami. W trybie tekstowym pracy, obrazy na monitorach graficznych wymagały ręcznej aktualizacji. Jeśli więc użytkownik pomylił się przy tworzeniu kompozycji mapy, całą operację musiał powtarzać od początku.

Tryb GUI znacznie ułatwił takie operacje, automatyzując aktualizacje i odświeżanie monitorów graficznych.

W roku 2006 osoby, zajmujące się rozbudową programu GRASS, zdecydowały o zmianie narzędzi do budowy GUI.



Rys. 3.3 Graficzny interfejs użytkownika programu GRASS 6.4 w trybie tcltk.



Rys. 4 Graficzny interfejs użytkownika programu GRASS 6.4 w trybie wxPython.

W pierwszej kolejności należało podjąć decyzję o języku programowania. Optymalny język, według programistów projektu, powinien spełniać następujące kryteria [4]:

- łatwy i szybki do nauki,
- współpracujący z większością

bibliotek graficznych,
- popularny i nowoczesny,
- udostępniający funkcje niskopoziomowe,
- zorientowany obiektowo.

Ostatecznie zdecydowano o wyborze języka Python. Pozostała jeszcze kwestia wyboru środowiska graficznego. Rozważano wybór jednej z trzech opcji [4]: Gtk, Qt lub wxWidgets/wxPython. Biorąc pod uwagę warunki licencjonowania oraz inne czynniki, zdecydowano o wyborze środowiska wxPython. Pracę nad nową wersją GUI rozpoczęli Michael Barton, Daniel Calvano, Martin Landa, Jachym Cepicky oraz Glynn Clements. W chwili obecnej głównym programistą projektu pracującym nad rozwojem GUI GRASS jest Martin Landa z Czech [3].

Aby uruchomić program GRASS z interfejsem graficznym wxPython należy wydać komendę

```
grass64 –wxpython
```

Choć wygląd GUI-wxPython (rys. 3.4) nieznacznie tylko odbiega od wyglądu GUI-tcltk (rys. 3.3), zmiana środowiska ułatwiła stworzenie pierwszej wersji programu GRASS nie tylko dla systemów Unix i MacOS, ale również pracującej w systemie Windows i nie wykorzystującej bibliotek pomocniczych takich jak CygWin. Chociaż z tego jednego powodu zmiana GUI była warta zachodu. Wersja ta jest w trakcie rozwoju i pełną funkcjonalność powinna uzyskać wraz z wydaniem GRASS 7.0.

3.4. Ujednolicenie interfejsów graficznych

Open Source Geospatial Foundation (OSGeo) sprawuje opiekę nad rozwojem wolnego oprogramowania geomatycznego. Jednym z projektów realizowanych w ramach fundacji jest OSGeo Graphics [7]. Jego celem jest stworzenie wspólnej biblioteki ikon dla programów geomatycznych. W chwili obecnej, pod nazwą GIS icons, udostępnianych jest na wolnej licencji ponad 200 ikon [5] związanych z systemami informacji przestrzennej (rys. 3.5).

Ikony te powstawały przy ścisłej współpracy z zespołami programistów GRASS oraz Quantum GIS. Stały się bazowym motywem ikon w najnowszych wydaniach tych dwóch programów (rys. 3.6). Łatwiejsze powinno stać się zatem korzystanie z tych dwóch programów dla początkujących użytkowników.

Do stworzenia ikon wykorzystano wolne i otwarte oprogramowanie – Inkscape [2], a założenia graficzne przyjęto w oparciu o wytyczne takich firm jak Apple [1] i Microsoft [6] oraz projektu Tango [8].

3.5. Dyskusja i wnioski

Dla osób, które rozpoczynają pracę z programem GRASS, wykorzystanie GUI powinno stanowić duże ułatwienie. Szczególnie jeśli będzie to interfejs zbliżony wyglądem do podobnych, znanych już programów.

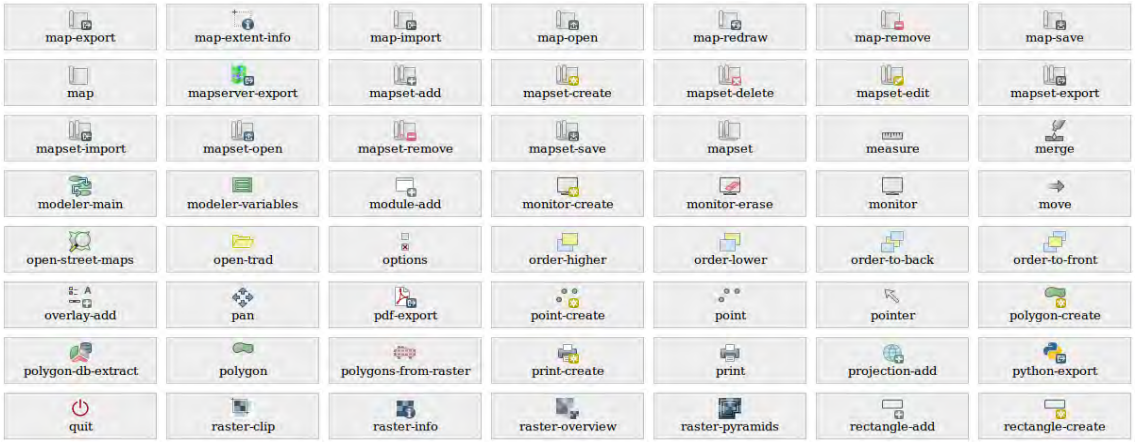
Są jednak sytuacje, w których praca w trybie tekstowym staje się koniecznością. Należy pamiętać o tym, że GUI prawie nigdy nie udostępnia wszystkich możliwości systemu. I zasada ta nie dotyczy wyłącznie programu GRASS. Nie zawsze interfejs graficzny GRASS udostępnia komplet opcji danego modułu. Warto więc sięgnąć do dokumentacji modułu i sprawdzić czy aby takie ograniczenie nie jest dla nas przeszkodą.

System GRASS ma budowę modułową i wiele bardziej skomplikowanych operacji może być realizowanych poprzez skrypty wywołujące poszczególne moduły. A to jest nadal domena komunikacji terminalowej. W tym tkwi chyba największa siła GRASS-a.

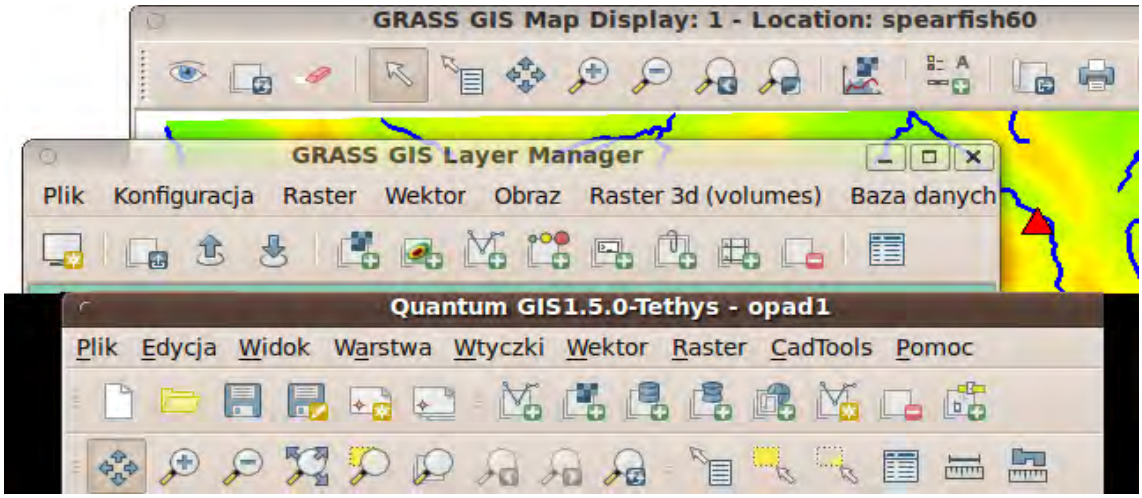
Ponieważ GRASS jest bardzo rozbudowanym programem, czasem odszukanie komendy w menu lub na pasku narzędzi może być czasochłonne. Osoby pracujące dłużej z programem, szybciej uruchomią wiele komend wpisując je w trybie tekstowym lub wywołując historię komend. Należy pamiętać, że w trybach graficznych również możemy

uruchamiać komendy poleceniami wydawanymi z klawiatury. Trzeba też wiedzieć o tym, że nie wszystkie dostępne moduły znajdują się w menu.

GRASS GIS jest wygodnym i ergonomicznym systemem, ponieważ nie ogranicza użytkowników. Każdy z nich może pracować z takim interfejsem, który najbardziej mu odpowiada.



Rys. 3.5 Przykładowe ikony z motywu QGIS icons.



Rys. 3.6 Ujednolicone interfejsy graficzne GRASS 6.4 i QGIS 1.5 wykorzystujące motyw QGIS icons.

Źródła internetowe

[1] Apple, Designing toolbar icons, Apple Human Interface Guidelines, http://developer.apple.com/documentation/UserExperience/Conceptual/AppleHIGuidelines/
[2] Inkscape project, 2010, http://inkscape.org/
[3] M.Landa, 2007, GUI development for GRASS GIS. Geoinformatics FCE CTU 2007, Workshop Proceedings Vol. 2, Prague
[4] http://les-ejk.cz/2007/05/wxgrass-new-grass-gui-1/
[5] OSGeo Graphics project, 2010, http://robert.szczepanek.pl/osgeo-graphics/toolbar-icons/24x24/
[6] Microsoft, 2001, Creating Windows XP Icons, Windows XP Technical Articles, http://msdn.microsoft.com/en-us/library/ms997636.aspx

[7] R.Szczepanek, 2008, Toolbar icons for GIS applications, Geoinformatics FCE CTU, Volume 3, ISSN 1802-2669, Czech Technical University in Prague
[8] Tango Desktop Project, http://tango.freedesktop.org/Tango Desktop Project

4. Moduł r.sun – wykorzystanie do obliczania wydajności kolektorów słonecznych

Małgorzata Pietras

4.1. Wstęp

Znaczenie energii słonecznej jako alternatywnego źródła energii wzrasta. Jest to spowodowane zarówno wyczerpywaniem się tradycyjnych nośników energii, jak i rosnącymi cenami energii.

Na problem zbyt intensywnego wyczerpywania zasobów przyrody jako pierwszy na forum międzynarodowym zwrócił uwagę Sekretarz Generalny ONZ U'Thant. Raport U'Thanta zatytułowany „Człowiek i jego środowisko" został ogłoszony na sesji Zgromadzenia Ogólnego w dniu 26 maja 1969 roku.

Od czasów raportu U'Thanta minęło 40 lat. Przez ten czas udało się opracować bardziej efektywne technologie wykorzystania tradycyjnych źródeł energii, jednak zapotrzebowanie na energię wciąż rośnie. Dlatego właśnie szczególną uwagę zwraca się obecnie na oszczędność energii oraz możliwości jej pozyskania ze źródeł odnawialnych. W tym aspekcie ważna jest także kwestia zmniejszania emisji zanieczyszczeń atmosferycznych.

Korzyści jakie płyną z wykorzystywania odnawialnych źródeł energii (OZE) sprawiły, że to zagadnienie zostało ujęte i zdefiniowane w Dyrektywie Parlamentu Europejskiego i UE nr 96/92/WE z dnia 19 grudnia 1996r. W sprawie wspólnych zasad dla wewnętrznego rynku energii elektrycznej [9]. Dyrektywa wyszczególnia następujące źródła energii odnawialnej: wiatr, słońce, ciepło geotermalne, ruchy morza, biomasa, instalacje wodne o możliwości wytwarzania energii do 10 MW. W wykładni prawa Wspólnoty Europejskiej, powyższe źródła stanowią zasoby dostarczające energię. Dyrektywa UE zakłada, że do końca 2010 roku źródła alternatywne (OZE) będą dostarczały 15% ogółu wykorzystywanej energii. Polskie regulacje prawne, bazując na ustawie – prawo energetyczne, zakładają do końca 2010 roku udział OZE na poziomie 7,5%, a w 2020 do 14%.

W Polsce polityka państwa w zakresie promocji odnawialnych źródeł energii jest określona przez przepisy zawarte w ustawie Prawo energetyczne i w rozporządzeniach do tej ustawy. Dokument ten został przyjęty przez Radę Ministrów w dniu 4 stycznia 2005 roku pod nazwą „Polityka energetyczna Polski do 2025 roku". Określa ona cele polityki energetycznej państwa do roku 2025, ze szczególnym naciskiem na następujące cele:

- zapewnienie bezpieczeństwa energetycznego kraju,
- wzrost konkurencyjności gospodarki i jej efektywności energetycznej,
- ochrona środowiska przed negatywnymi skutkami działalności energetycznej związanej z wytwarzaniem, przesyłaniem i dystrybucją energii i paliw [5].

W celu podniesienia atrakcyjności zastosowania energii wytworzonej z OZE w Polsce wdrażane są liczne programy mające na względzie opłacalność pozyskiwania energii ze źródeł odnawialnych. Przedsiębiorstwa i samorządy mogą liczyć na dofinansowanie przy wprowadzaniu nowych, ekologicznych rozwiązań.

Szczególną funkcję pełni Program Operacyjny Infrastruktura i Środowisko, stanowiący cześć Narodowych Strategicznych Ram Odniesienia na lata 2007-2013 (NSRO) [7].

Powyższe dokumenty podkreślają znaczenie energii odnawialnej. Energia promieniowania słonecznego stanowi zarówno

odnawialne źródło energii, jak i źródło energii przyjaznej środowisku, ze względu na brak emisji zanieczyszczeń atmosferycznych.

Promieniowanie słoneczne stanowi niewyczerpalny zasób energii. Słońce promieniuje z podobną wydajnością od ponad pięciu miliardów lat. Ilość energii emitowanej przez Słońce szacuje się na około $3.846 \cdot 10^{26} W/s$. Jednak do powierzchni kuli ziemskiej dociera niewielka jej część [1].

Miarę ilości docierającego do Ziemi promieniowania słonecznego stanowi stała słoneczna. Jest to średnie natężenie promieniowania słonecznego ustalone dla górnej granicy atmosfery. Wartość stałej słonecznej podawana w literaturze wynosi 1325 W/m^2 [8]. Stała słoneczna stanowi parametr zmienny w czasie. Jest to spowodowane zmienną odległością Ziemi od Słońca oraz zmienną aktywnością Słońca.

Natężenie promieniowania słonecznego docierające do powierzchni Ziemi jest mniejsze od stałej słonecznej, bowiem podczas przejścia przez atmosferę promieniowanie ulega osłabieniu. Osłabienie to jest spowodowane przez zawarte w atmosferze cząstki gazów, aerozoli oraz pary wodnej.

Do powierzchni Ziemi promieniowanie słoneczne dociera w postaci promieniowania bezpośredniego oraz rozproszonego. Promieniowaniem bezpośrednim nazywamy tę część promieniowania słonecznego, która dociera do powierzchni Ziemi w postaci wiązki promieni równoległych. Promieniowanie rozproszone natomiast dociera do powierzchni Ziemi w wiązkach nieregularnych. Suma promieniowania bezpośredniego i rozproszonego to promieniowanie całkowite czyli całkowita ilość energii słonecznej przypadająca na metr kwadratowy powierzchni.Ponieważ wartość natężenia promieniowania całkowitego docierającego do danego miejsca zależy od ukształtowania terenu, zachmurzenia, zanieczyszczeń atmosferycznych oraz współrzędnych geograficznych danego punktu, przy wielu rozważanych parametrach, dobre rozwiązanie stanowią modele pozwalające na szybkie i efektywne uwzględnienie wszystkich tych czynników.

Celem poniższych analiz jest przedstawienie wstępnych wyników zastosowania modułu r.sun systemu GRASS do określania przestrzennego zróżnicowania dopływu potencjalnego promieniowania słonecznego dla wybranego obszaru oraz wydzielenie kompleksów zabudowy korzystnych pod względem wykorzystania energii promieniowania słonecznego do wytwarzania cieplej wody użytkowej (c.w.u).

Jak już wspomniano, aby efektywnie wykorzystywać energię słoneczną należy dobrze rozpoznać jej zasoby. Do tej pory oceniając zasoby helioenergetyczne Polski brano pod uwagę ilość godzin ze słońcem, sumy miesięczne i roczne promieniowania, przezroczystość atmosfery oraz lokalne zróżnicowanie rzeźby terenu. Na podstawie tych parametrów wydzielono jedenaście regionów uszeregowanych pod kątem przydatności dla potrzeb energetyki solarnej [1]. Zarówno w tym, jak i w podobnych opracowaniach, nie uwzględniano zacienienia wywołanego przez zabudowę oraz tereny zielone. Warto także podkreślić, iż wcześniejsze opracowania skupiały się na analizie większych obszarów i wydzielaniem w nich stref, dla których przyjmowano wartości średnie natężenia promieniowania słonecznego. Jednak, jeżeli cel analizy ma stanowić ocena efektywności wykorzystania promieniowania słonecznego jako źródła energii, zasadne staje się dokładne ocenienie jej zasobów.

W odróżnieniu od wcześniejszych opracowań podjęto próbę uwzględnienia wpływu zacienienia spowodowanego przez zabudowę oraz zieleń na wielkość ilości docierającego promieniowania. Analiza wykorzystująca moduł r.sun programu GRASS daje możliwość ocenienia efektywności stosowania kolektorów słonecznych dla pojedynczych budynków, zamiast dla całych regionów.

4.2. Narzędzia

Jako narzędzie w analizie zastosowano moduł r.sun [4], który stanowi część systemu informacji przestrzennej GRASS.

Moduł r.sun umożliwia obliczenie parametrów takich jak:

- natężenie promieniowania bezpośredniego,
- natężenie promieniowania odbitego,
- natężenie promieniowania rozproszonego.

Wielkość natężenia promieniowania słonecznego podawana jest w W/m^2, natomiast sumy natężenia promieniowania słonecznego w Wh/m^2.

Danymi wejściowymi do obliczeń jest cyfrowy model terenu. Dla modelu terenu przy użyciu narzędzia r.slope.aspect tworzone są warstwy nachyleń i ekspozycji stoków na zadanym obszarze.

Kolejny parametr wejściowy stanowi współczynnik zmętnienia Linkego. Współczynnik zmętnienia Linkego opisuje zmniejszenie promieniowania wywołane wspólnym działaniem aerozoli atmosferycznych i pary wodnej. Do jego wyliczenia zastosowana została formuła empiryczna zaproponowana przez Dougniaux [2]:

$$LTF = \left(\frac{85 + \gamma}{39.5 \cdot e^{-PCW} + 47.4} + 0.1 \right) \cdot (16 + 0.22 \cdot PWC) \cdot \beta_A$$

gdzie:

γ – wysokość słońca w stopniach,

PWC – ciśnienie pary wodnej,

β_A – wskaźnik Angstroma wyrażający zawartość aerozoli w atmosferze.

W module r.sun współczynnik zmętnienia Linkiego jest domyślnie przyjmowany jako wartość średnia wynosząca 3.0. Wartość ta stanowi średnią roczną dla obszarów zurbanizowanych [4]. Parametr albedo, wykorzystywany przez moduł r.sun, przyjmuje domyślną wielkość 0.2, natomiast przyjmowana domyślna wielkość stałej słonecznej wynosi 1367 W/m^2. Wielkości domyślne parametrów współczynnika zmętnienia Linkego oraz albedo mogą zostać zdefiniowane w momencie uruchamiania modułu r.sun. Mogą one stanowić zarówno wartość średnią dla analizowanego obszaru lub warstwę rastrową zawierającą wielkości tych parametrów dla analizowanego obszaru.

W zbiorze danych wejściowych znajdują się także współrzędne geograficzne danego punktu oraz, opcjonalnie, mapy rastrowe z wielkościami współczynnika rzeczywistego promieniowania bezpośredniego i rozproszonego. W procedurze obliczeń określić należy także dzień, dla którego mają być policzone wartości oraz rozdzielczość rastra prowadzonych obliczeń.

Dane wyjściowe to wynikowa warstwa rastrowa natężenia potencjalnego promieniowania bezpośredniego, rozproszonego, odbitego od powierzchni ziemi oraz długości czasu trwania usłonecznienia. Wyniki stanowią średnie dobowe sumy promieniowania dla danej rozdzielczości rastra.

W celu uwzględnienia w obliczeniach zacienienia, cyfrowy model terenu poszerzono o informacje o wysokości budynków i obszarów zielonych. Informacja ta pozwoliła na uwzględnienie efektu zacienienia wywołanego zabudową oraz zielenią. Możliwość taką stwarza opcja „-s” modułu r.sun.

4.3. Teren badań

Do analizy wybrano osiedle Dąbie. Osiedle Dąbie położone jest we wschodniej części Wrocławia. Zabudowę osiedla stanowią wielorodzinne dwukondygnacyjne domy szeregowe, cztero- lub sześcio-mieszkaniowe, domy wolno stojące oraz pewna liczba domów jednorodzinnych. Poza zabudową, znaczną cześć osiedla stanową parki i inne publiczne tereny zielone.

Rys. 4.1 Obszar wybrany do analizy. Osiedle Dąbie, Wrocław. Zaznaczono lokalizację testowych budynków oraz Obserwatorium Zakładu Klimatologii i Ochrony Atmosfery we Wrocławiu.

O wyborze właśnie tej części miasta wpływ miała zabudowa oraz występowanie sporej ilości terenów pokrytych wysoką i niską zielenią. Osiedle to bowiem bardzo dobrze reprezentuje typowe osiedla domów jednorodzinnych, a właśnie właściciele takich nieruchomości najczęściej korzystają z instalacji solarnych. Dodatkowy atut osiedla to obecność urozmaiconej pokrywy roślinnej. Znajdujemy tu zarówno niską przydomową zieleń (do 8 metrów) oraz zieleń miejską wysoką (do 18 metrów). Wpływa to na ilość docierającego promieniowania słonecznego do powierzchni dachów domów.

Zasięg przestrzenny analizowanego obszaru przedstawiono na rysunku 4.1.

4.4. Obliczenia sum promieniowania

Praca z modułem r.sun może przebiegać zarówno w środowisku interfejsu graficznego systemu GRASS, jak i z linii poleceń.

Aby uruchomić r.sun przy pomocy interfejsu graficznego wybieramy z menu podmenu „raster” a następnie pozycję „promieniowanie słoneczne i cienie” (w wersji polskiej menu). Kolejny krok to podanie poszczególnych parametrów niezbędnych do obliczeń (tj. podanie nazwy wcześniej przygotowanej warstwy nachyleń i ekspozycji, warstwy zawierającej cyfrowy model terenu, numeru dnia w roku itd.) oraz podanie wielkości jaką obliczamy i nazwy warstwy wynikowej. Następnie uruchamiamy model.

Korzystając z linii komend należy wszystkie parametry podać w poleceniu:

```
r.sun elevin=nazwa aspin=nazwa slopei
   n=nazwa[ linkein=nazwa]
   [ lin=wartość] [ albedo=nazwa]
   [ alb=wartość] [ latin=nazwa]
   [ lat=wartość] [ coefbh=nazwa]
   [coefdh=nazwa] [ incidout=nazwa]
   [ beam_rad=nazwa]
   [ insol_time=nazwa]
   [ diff_rad=nazwa]
   [ refl_rad=nazwa] day=wartość
   [ step=wartość] [ declin=wartość]
   [time=wartość] [dist=wartość]
```

gdzie:

- elevin – nazwa warstwy rastrowej zawierającej wysokości terenu używana jako wejście,
- slopein - nazwa warstwy rastrowej z wiekościami nachyleń,
- aspin – nazwa warstwy rastrowej z wielkościami ekspozycji,

- linkein – nazwa warstwy z wielkością współczynnika zmętnienia Linkego,
- lin – wielkość współczynnika zmętnienia Linkego (domyślnie 3.0)
- albedo – nazwa warstwy rastrowej z wielkością albedo,
- alb – wielkość albedo (domyślnie 0.2),
- latin - nazwa warstwy rastrowej z określoną szerokością geograficzną (stopnie dziesiętnie),
- lat – szerokość geograficzna,
- coefbh – nazwa warstwy rastrowej z wielkością rzeczywistego współczynnika promieniowania bezpośredniego,
- coefdh - nazwa warstwy rastrowej z wielkością rzeczywistego współczynnika promieniowania rozproszonego,
- incidout – nazwa warstwy wyjściowej z wielkościami kąta padania promieni słonecznych,
- insol_time – nazwa warstwy wyjściowej zawierającej długość trwania usłonecznienia,
- beam_rad – nazwa warstwy wyjściowej z wielkością promieniowania bezpośredniego,
- diff_rad - nazwa warstwy wyjściowej z wielkością promieniowania rozproszonego,
- refl_rad – nazwa warstwy wyjściowej z wielkością promieniowania odbitego od powierzchni ziemi,
- day – numer dnia w roku (od 1 do 365),
- step – krok czasowy, w którym liczone są dzienne sumy promieniowania (domyślnie 0.5h),
- declin – wielkość deklinacji,
- time – czas według lokalnego czasu słonecznego,
- dist – rozdzielczość przestrzenna (domyślnie 1.0).

Ze względu na fakt, iż uzyskane wyniki stanowią sumy natężenia promieniowania dla konkretnych dni, chcąc uzyskać dane o dłuższym okresie czasu najbardziej efektywnym rozwiązaniem jest posłużenie się skryptem. Taka operacja znacznie skraca czas dokonywanych obliczeń. Poszczególne dni pogrupowane zostały w miesiące, następnie policzono sumy natężenia promieniowania dla poszczególnych miesięcy. Informację o wartości promieniowania całkowitego uzyskano poprzez operację sumowania warstw promieniowania bezpośredniego i całkowitego. Moduł r.sun uruchomiony został z ustawieniami domyślnymi dla współczynnika zmętnienia Linkego oraz albedo.

Poniżej przedstawione są skrypty realizujący opisane funkcje:

- skrypt wykonujący obliczenia dla całego roku „prom_rok_suma.sh”

```
#!/bin/sh

echo "Miesiąc: 01"
dni=$(awk 'BEGIN {for(i=1;i<32;i++)
   print i;}')
sh prom_suma.sh beam_01 diff_01
   "$dni"

echo "Miesiąc: 02"
dni=$(awk 'BEGIN {for(i=32;i<60;i++)
   print i;}')
sh prom_suma.sh beam_02 diff_02
   "$dni"

echo "Miesiąc: 03"
dni=$(awk 'BEGIN {for(i=60;i<91;i++)
   print i;}')
sh prom_suma.sh beam_02 diff_03
   "$dni"

echo "Miesiąc: 04"
dni=$(awk 'BEGIN {for(i=91;i<121;i++)
   print i;}')
sh prom_suma.sh beam_04 diff_04
   "$dni"

echo "Miesiąc: 05"
dni=$(awk 'BEGIN {for(i=121;i<152;
   i++)print i;}')
sh prom_suma.sh beam_05 diff_05
   "$dni"

echo "Miesiąc: 06"
dni=$(awk 'BEGIN {for(i=152;1<182;
   i++)print i;}')echo "Miesiąc: 01"
dni=$(awk 'BEGIN {for(i=1;i<32;i++)
   print i;}')
sh prom_suma.sh beam_01 diff_01
   "$dni"

echo "Miesiąc: 07"
dni=$(awk 'BEGIN {for(i=182;i<213;
   i++) print i;}')
sh prom_suma.sh beam_07 diff_07
   "$dni"

echo "Miesiąc: 08"
dni=$(awk 'BEGIN {for(i=213;i<244;i++)
   print i;}')
sh prom_suma.sh beam_08 diff_08
   "$dni"

echo "Miesiąc: 09"
dni=$(awk 'BEGIN {for(i=244;i<274;i++)
   print i;}')
sh prom_suma.sh beam_09 diff_09
   "$dni"

echo "Miesiąc: 10"
```

```
dni=$(awk 'BEGIN {for(i=274;i<305;
   i++) print i;}')
sh prom_suma.sh beam_10 diff_10
   "$dni"
echo "Miesiąc: 11"
dni=$(awk 'BEGIN {for(i=305;i<335;
   i++) print i;}')
sh prom_suma.sh beam_11 diff_11
   "$dni"

echo "Miesiąc: 12"
dni=$(awk 'BEGIN {for(i=235;i<336;
   i++) print i;}')
sh prom_suma.sh beam_12 diff_12
   "$dni"
```

- skrypt wykonujący obliczenia dla wybranego okresu „prom_suma.sh”

```
#!/bin/sh

sumabeam=$1
sumadiff=$2
r.mapcalc "$sumabeam=0.0"
r.mapcalc "$sumadiff=0.0"
for dzien in $3
do
   echo "Dzień $dzien"
   diff="dzien_diff_nr_$dzien"
   beam="dzien_beam_nr_$dzien"
   r.sun –s –overwrite
     elevin=dabie_dem
     aspin=dabie_ekspozycja
     slopein=dabie_nachylenia
   d.erase
   d.rast $beam
   r.mapcalc "$sumadiff=$sumadiff+
     $diff"
   r.mapcalc "$sumabeam=$sumabeam+
     $beam"
done
```

4.5. Analiza wyników obliczeń

Cyfrowy model wysokościowy SRTM poszerzono o informacje o wysokości budynków oraz zieleni. No podstawie tych danych, uzyskano za pomocą modułu r.sun systemu GRASS przestrzenny rozkład promieniowania całkowitego dla wybranego obszaru Wrocławia. Rozkład ten przedstawiono na rysunku 4.2.

Posługując się przestrzennym rozkładem promieniowania słonecznego uwzględniającym zacienienia wywołane przez budynki i obszary zielone, możliwe staje się dokładniejsze wyznaczenie obszarów bardziej i mniej korzystnych pod względem użytkowania kolektorów słonecznych. Projektując instalacje solarne zwykło posługiwać się danymi meteorologicznymi opisującymi natężenie promieniowania oraz temperaturę powietrza. Dane te często pochodzą ze stacji meteorologicznych położonych w odmiennych warunkach lokalnych niż analizowany obszar. Jest to istotne szczególnie ze względu na otoczenie przez zieleń oraz zabudowę. Zatem dane te często nie opisują dokładnie warunków meteorologicznych wybranego miejsca.

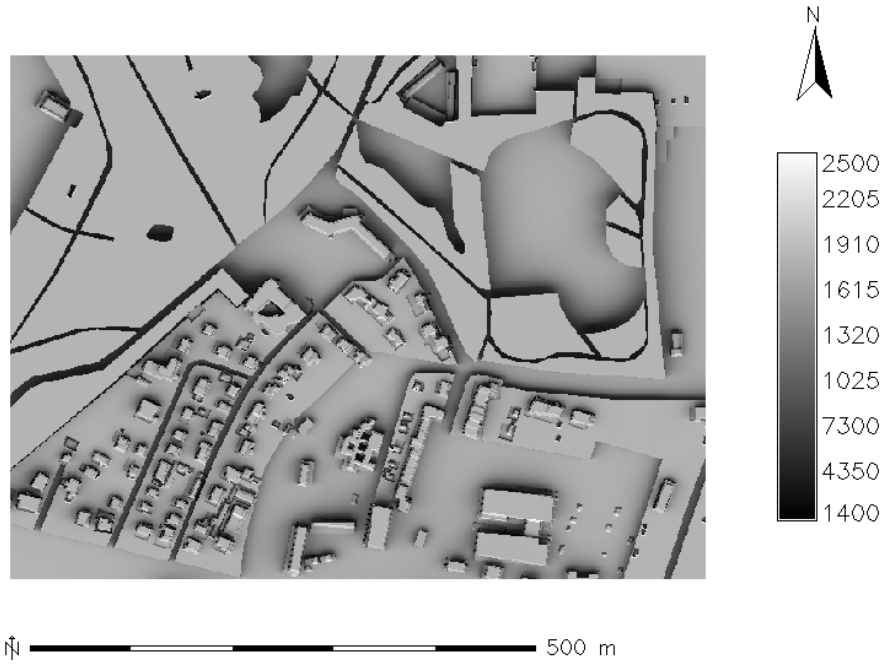
Korzystając z wyników otrzymanych przy pomocy modułu r.sun można natomiast uzyskać wartości natężenia promieniowania słonecznego dla danego miejsca (lokalizacji instalacji solarnej). Ze względu na to, że instalacje solarne są zakładane na dachach zabudowań można pominąć błędy obliczeniowe r.sun wynikające z rastryzacji wejściowego modelu wysokościowego, pojawiające się na brzegach budynków (na ścianach).

Wiarygodność obliczeń uzyskanych przez zastosowanie modelu r.sun została zweryfikowana poprzez zestawienie wyników modelowych z danymi pochodzącymi z Obserwatorium Zakładu Klimatologii i Ochrony Atmosfery Uniwersytetu Wrocławskiego. Pomiary promieniowania całkowitego oraz rozproszonego wykonywane są przez stację automatyczną z krokiem minutowym.

Różnice te sięgają 100kWh/m^2 miesięcznie dla lipca, co stanowi zawyżenie o 64.5% w stosunku do wielkości zmierzonej. Występowanie różnic wskazuje na nieprawidłowości leżące po stronie modelu. Jednak, mimo to widoczna jest duża różnica pomiędzy punktem A i B. Widać, iż punkt A zacieniony przez zieleń wysoką rocznie otrzymuje znikomą ilość promieniowania słonecznego. Ilość ta odpowiada wielkości promieniowania rozproszonego bowiem do tak zacienionego obszaru dociera jedynie promieniowanie rozproszone.

Na podstawie wyników pomiarów stacji automatycznej, obliczono średnie sumy roczne i miesięczne promieniowania słonecznego całkowitego i rozproszonego dla lat 1999-2005.

Zestawienie sum rocznych promieniowania słonecznego całkowitego wyliczonego przed moduł r.sun z danymi z pomiarów aktynometrycznych (tabela 4.1) pokazuje zawyżanie przez r.sun wyliczonych wielkości promieniowania słonecznego.



Rys. 4.2 Przestrzenny rozkład sumy rocznej promieniowania całkowitego [kWh/m^2] dla wybranego obszaru Wrocławia z uwzględnieniem zacienienia przez budynki i obszary zielone.

Tabela 4.1 Średnie sumy miesięczne promieniowania całkowitego [kWh/m^2] we Wrocławiu w latach 1999-2005 zmierzone oraz obliczone przy użyciu modelu r.sun

Mies.	Wielkość zmierzona	Wielkość wyliczona przez r.sun		
		Punkt A	Punkt B	Obserwatorium
I	37.20	13.20	44.20	44.00
II	48.80	9.76	38.60	38.30
III	90.00	26.80	138.00	135.50
IV	133.60	32.70	193.60	159.00
V	185.70	37.70	239.00	238.90
VI	176.60	40.80	268.30	268.30
VII	156.20	39.60	257.00	257.00
VIII	157.20	35.50	216.00	216.10
IX	103.10	28.30	152.90	152.90
X	69.60	21.80	97.90	97.50
XI	40.00	14.30	51.70	51.50
XII	30.20	11.40	36.10	35.50
IV-X	982.00	236.40	1424.70	1389.70
XI-III	246.20	75.46	308.60	304.80

Największe rozbieżności występują w miesiącach ciepłej pory roku (od marca do września). W zestawieniu sum natężenia promieniowania dla ciepłej (IV-X) i chłodnej pory (XI-III) roku również widoczne jest zawyżanie wielkości natężenia promieniowania przez model. Porę ciepłą charakteryzuje większa różnica wynosząca 407.7 kWh/m^2, co stanowi 41.5% wielkości zmierzonej. W porze chłodnej różnica miedzy wartościami zmierzonymi a obliczonymi przez r.sun dla punktu reprezentującego obserwatorium jest mniejsza i wynosi 57.8 kWh/m^2 (19% wielkości zmierzonej). Wielkości sum promieniowania słonecznego w porze chłodnej są szczególnie istotne z punktu widzenia rozważań na temat efektywności pracy instalacji solarnych, bowiem wówczas zapotrzebowanie na ciepło wzrasta a dodatkowo sprawność pracy instalacji jest obniżana przez niską temperaturę otoczenia.

4.6. Obliczanie wydajności kolektora słonecznego

W projektowaniu instalacji solarnych najczęściej stosowana jest metoda F-CHART.

Metoda F-CHART jest powszechnie stosowana w projektowaniu instalacji solarnych. Pozwala ona na szacunkową ocenę stopnia pokrycia potrzeb cieplnych przez energię słoneczną. Metoda ta wymaga wprowadzenia danych meteorologicznych: temperatury powietrza oraz sumy promieniowania całkowitego. Uwzględnia ona zarówno warunki meteorologiczne panujące wokół instalacji, jak i parametry techniczne samej instalacji. Stosując metodę zakłada się:

- sumaryczna powierzchnia pochłaniająca kolektorów: S = 3.5m^2 (typowe płaskie kolektory z powierzchnią selektywną);
- pojemność zbiornika akumulacyjnego: 350 litrów;
- wymagana temperatura wody ciepłej: 45°C;
- temperatura wody zasilającej (wodociągowej): zmienna od 10°C zimą do 15°C latem;
- współczynnik α równy jest 0.8 (maksymalna sprawność kolektorów);
- średni współczynnik strat ciepła z kolektora do otoczenia F_r=3;
- kolektory pochylone są pod kątem 45° względem poziomu i skierowane na południe;
- zużycie ciepłej wody wnosi 300 litrów na dobę [10, 11].

Posługując się metodą F-Chart oraz danymi meteorologicznymi pochodzącymi z Obserwatorium Zakładu Klimatologii i Ochrony Atmosfery Uniwersytetu Wrocławskiego [3], obliczono pokrycie zapotrzebowania na c.w.u. Analizy przeprowadzono dla okresu 1999-2005 [6] dla poszczególnych lat, sezonów oraz miesięcy. Następnie, posługując się tą samą metodą, dokonano obliczeń dla tego samego okresu przyjmując wartości natężenia promieniowania słonecznego obliczone za pomocą modułu r.sun. Do obliczeń wybrano dwa budynki (A i B) różnie zasłaniane przez zieleń wysoką oraz budynek Obserwatorium Zakładu Klimatologii i Ochrony Atmosfery.

Wyniki analizy zestawiono w tabelach 4.2 i 4.3. Widoczna jest duża, ponad pięciokrotna, różnica w ilości ciepła dostarczonego z kolektora słonecznego między punktem A i B. Różnica w ilości dostarczanego ciepła przez kolektor pracujący na dachu budynku A i B spowodowana jest odmienną lokalizacją obu obiektów. Budynek A znajduje się bowiem pośrodku obszaru porośniętego zielenią wysoką. Natomiast budynek B otoczony jest przez zieleń niską, która nie powoduje zacienienia dachu budynku.

Dodatkowo przedstawiono wyniki analizy dla punktu reprezentującego położenie Obserwatorium Zakładu Klimatologii i Ochrony Atmosfery Uniwersytetu Wrocławskiego oraz wykorzystującej rzeczywiste dane zmierzone przy pomocy aktynometrów w obserwatorium meteorologicznym.

Prognozowaną ilość ciepła możliwą do uzyskania w danym punkcie określa współczynnik f (f=1 oznacza 100% pokrycie zapotrzebowania na c.w.u). Współczynnik ten jest parametrem kompleksowym, uwzględnia bowiem zarówno parametry meteorologiczne jak natężenie promieniowania i temperaturę otoczenia, jak również straty ciepła spowodowane charakterystyką pracy instalacji.

Analiza wielkości współczynnika f dla poszczególnych miesięcy zarówno dla danych aktynometrycznych, jak i wartości wygenerowanych przy użyciu r.sun (rys. 4.3) wskazuje, iż różnice miedzy wartościami natężeniu promieniowania słonecznego miedzy danymi rzeczywistymi a pochodzącymi z modelu mają niewielkie przełożenie na ilość uzyskiwanego ciepła.

Różnice pomiędzy wielkością współczynnika f są tutaj niewielkie. Bardzo niskie wartości współczynnika f dla punktu A wskazują jednoznacznie na duży wpływ zacienienia na efektywność pracy instalacji solarnej.

Tabela 4.2 Średnia suma roczna promieniowania całkowitego [kWh/m^2] we Wrocławiu w latach 1999-2005

Wielkość zmierzona w Obserwatorium ZKiOA UWr	
Obserwatorium (aktynometr)	1228.26
Wielkości uzyskane przy użyciu modelu r.sun	
Punkt A	311.86
Punkt B	1733.30
Obserwatorium	1694.50

Tabela 4.3 Roczne sumy ilości ciepła dostarczonego z instalacji obliczone dla danych z pomiarów aktynometrycznych i danych uzyskanych przy użyciu modelu r.sun

Wielkości zmierzone w Obserwatorium ZKiOA UWr	
Obserwatorium (aktynometr)	2919.66
Wielkości uzyskane przy użyciu modelu r.sun	
Punkt A	585.59
Punkt B	3646.50
Obserwatorium	3591.60

4.7. Wnioski

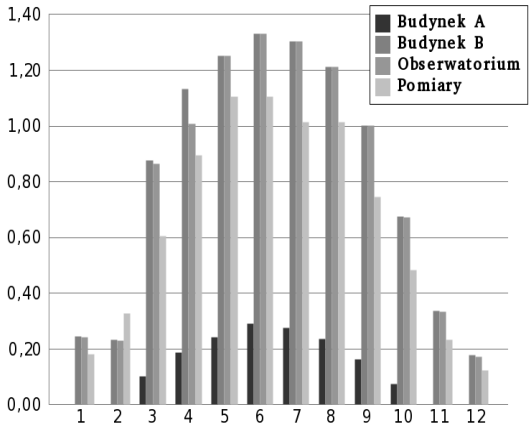
Opłacalność korzystania z kolektorów słonecznych w celu uzyskiwania ciepłej wody użytkowej w dużej mierze zależy od ich usytuowania. Użytkownikom w decyzji dotyczącej instalacji kolektora posłużyć mogą modele przestrzenne rozkładu promieniowania słonecznego.

Zaobserwowane różnice pomiędzy punktami o różnej lokalizacji są dla użytkowników kolektorów słonecznych cenną informacją.

Wpływ lokalizacji na stopień pokrycia zapotrzebowania na c.w.u jest szczególnie istotny w chłodnej porze roku. Wówczas bowiem stopień pokrycia zapotrzebowania na ciepłą wodę użytkową przez pracujący kolektor wynosi około 20-25% (rys. 4.3). Biorąc pod uwagę obniżenie wydajności kolektora wywołaną zacienieniem przez inne budynki oraz zieleń, właściciele niektórych posesji mogą spodziewać się znacznie niższych wartości od tych deklarowanych przez producenta instalacji.

Należy podkreślić, iż w opracowaniu zestawiono wyniki analizy dla przykładowej instalacji. Sama analiza wymagała użycia wielu uogólnień.

Na przykład w ocenie zacienienia spowodowanego przez zieleń, nie uwzględniono sezonowej zmienności zwartości koron drzew. Poszerzenie analizy o ten aspekt stanowi ciekawy wątek dla dalszych badań.



Rys. 4.3 Wielkość współczynnika f w poszczególnych miesiącach.

Podobny problem stanowi domyślna wielkość współczynnika Linkiego. Wielkość tego parametru nie jest stała w ciągu roku. Jej zmienność zależy głównie od prężności pary wodnej zawartej w atmosferze oraz od ilości aerozoli w powietrzu atmosferycznym. Rozbieżności między danymi z przeprowadzonych pomiarów aktynometrycznych a wynikami obliczeń modułu r.sun (tab. 4.1) wskazują na konieczność uwzględnienia różnych wielkości tego współczynnika. Stosowanie jednej wielkości (3.0) dla całego roku jest niewłaściwe.

Literatura

[1] Chorościn G., 2002, *Energia Słońca – zasoby i jej wykorzystanie*, [w:] II Sympozjum koła naukowego „Ekofilia": Odnawialne źródła energii, Politechnika Wrocławska filia w Jeleniej Górze, Jelenia Góra, s. 89-104

[2] Dougniaux R., 1984, *De `influence de l `estimation du facteur de trouble atmospherique sur l `évaluation du rayonnement solaire direct per ciel clair. Application aux données radiométriques de L `IRM a' Uccle.* Institut Royal Météorologique de Belgique (IRM), Miscellanea, Ser. C

[3] A. Dubicki, M. Dubicka, M. Szymanowski, 2002, *Klimat Wrocławia*, [w:] Informator Środowisko Wrocławia, Dolnośląska Fundacja Ekorozwoju, Wrocław, s. 9-26

[4] Hofierka J. Šuri M., 2002, *The solar radiation model for Open Source GIS: implementation and applications*, Proc of the Open Source Gis-GRASS users conference 2002 – Treto, Italy, 11-13 September 2002, s. 1-19

[5] Kozyra J, Krawczyk G., 2006, *Założenia wzrostu wykorzystania odnawialnych źródeł energii przyjęte w dokumencie „Polityka energetyczna Polski do 2025 roku*, [w:] Odnawialne źródła energii , Kalotka J. (red.), Instytut Technologii Eksploatacji – PIB, Radom, ss. 130

[6] Pietras M., 2009, *Ekologiczne i ekonomiczne aspekty wykorzystania energii słonecznej do pozyskiwania ciepłej wody użytkowej na przykładzie Wrocławia*, Ciepłownictwo Ogrzewnictwo Wentylacja, 04/2009, s. 18-19

[7] *Program Operacyjny Infrastruktura i Środowisko, Narodowe Strategiczne Ramy Odniesienia 2007-2013*, 2006, Dokument przyjęty przez Radę Ministrów 29 listopada 2006r, s. 1-165

[8] Woś A., 2000, *Meteorologia dla geografów*, PWN, Warszawa, ss. 324

[9] Zajdler R., 2002, *Odnawialne źródła energii w regulacjach Unii Europejskiej, Prawo Unii Europejskiej nr 5*, wrzesień – październik 2002r, s. 12-14

[10] Zawadzki M., 2005, *Instalacje solarne do produkcji ciepłej wody użytkowej w budownictwie jedno- i wielorodzinnym*, [w]: Ogrzewanie energią słoneczną – materiały konferencyjne, Polski Klub Ekologiczny, Wrocław, s. 13-21

[11] Zawadzki M., 2003, *Kolektory słoneczne pompy ciepła na tak*, Wydawnictwo Polska Ekologia, Warszawa, ss. 280

5. Obliczanie szorstkości terenu w mieście z wykorzystaniem systemu GRASS

Paweł Netzel, Jacek Ślopek

5.1. Wprowadzenie

W środowisku miejskim zmiany poczynione przez człowieka w naturalnym otoczeniu oddziałują na klimat na wiele sposobów. Modyfikacji ulegają np. pole temperatury, czy też pole wiatru spowodowane pojawieniem się przeszkód dla przemieszczającego się powietrza. Przeszkodami tymi na terenach miast są przede wszystkim budynki. Rozwój środowiska miejskiego pociąga za sobą konieczność określenia związków pomiędzy dynamicznie przekształcanym otoczeniem życia mieszkańców miast a klimatem lokalnym. Niemałą część gromadzonych danych stanowią cechy geometryczne obiektów znajdujących się w mieście. Znajomość pola powierzchni, wysokości, czy specyficznych cech poszczególnych elementów zabudowy czy zespołów zieleni pozwala na wyznaczenie współczynników branych pod uwagę w analizie wpływu terenów zurbanizowanych na klimat lokalny. Jednym z takich współczynników jest szorstkość terenu.

Dla warunków równowagi obojętnej zachodzi równość zwana prawem logarytmicznego profilu wiatru [8]:

$$\frac{\bar{u}(z)}{u^*} = k^{-1} \cdot \ln\left(\frac{z - z_d}{z_0}\right)$$

gdzie

k – stała von Karmana,

$\bar{u}(z)$ – średnia prędkość wiatru na wysokości z,

u^* – prędkość tarciowa,

z_d – wysokość płaszczyzny zerowej,

z_0 – długość szorstkości.

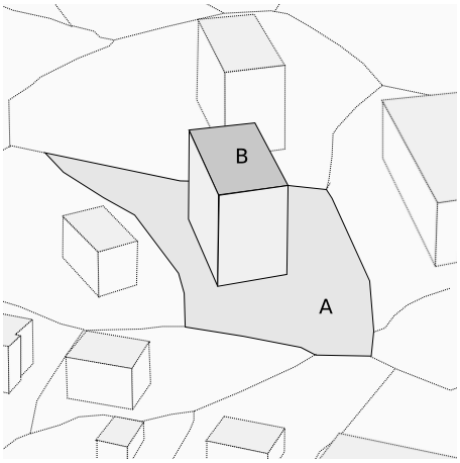
Parametry z_0 i z_d opisują szorstkość terenu. Długość szorstkości z_0 powiązana jest z przeciętnymi wysokościami przeszkód (krzewów, drzew, domów, trawy itp.) h_c. Zależy ona również od zmienności przestrzennej tych wysokości. Z grubsza przyjmuje się, że zachodzi zależność:

$$z_0 \sim 0.1 \cdot h_c \quad .$$

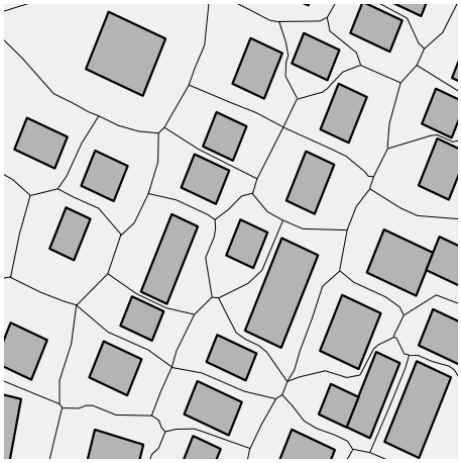
Znajomość tego parametru dla terenu zurbanizowanego, w powiązaniu z położeniem budynków (jak również z ich układami architektonicznymi) pozwala określić, jak bardzo zabudowa na danym obszarze będzie odpowiedzialna za wywoływanie zmian w kierunkach przepływu (lokalnej cyrkulacji powietrza), wzmacnianie efektów tunelowania przepływów, wywoływanie zmian w pionowym profilu prędkości wiatru, czy wytwarzanie lokalnych stref turbulencji. Nie pozostaje to bez wpływu na dyspersję zanieczyszczeń atmosferycznych, a także ma przełożenie na komfort życia na danym osiedlu, czy w dzielnicy. Parametry z_0 i z_d są potrzebne do określania współczynników równań modeli pozwalających na określanie zasięgu warstwy mieszania [1, 2, 9, 10, 11, 12, 13, 14]. Problem wygenerowania map szorstkości dla terenów miejskich podjęto w ramach projektu badawczego „Zróżnicowanie przestrzenne warstwy granicznej atmosfery na przykładzie Wrocławia i Krakowa”. Do analiz przyjęto teren miast Wrocław i Kraków wraz z otoczeniem obejmującym tereny podmiejskie.

5.2. Wysokość zabudowy jako źródło danych do określania szorstkości

Obliczenie współczynników szorstkości (z_0 i z_d) dla Wrocławia i Krakowa wykonano w oparciu o algorytmy zaimplementowane przez Gala, Sümeghy i Ungera [6, 7] dla węgierskiego miasta Szeged.



Pojedynczy obszar (A) przypisany do budynku (B).



Przykład przypisania obszarów dla obszaru zabudowanego.

Rys. 5.1 Wyznaczanie obszarów przypisywanych do budynków (lub ich grup), zgodne z podejściem Gala, Sümeghy i Ungera oraz Grimmonda i Oke'a.v

Gal i Unger w swojej pracy [7] zastosowali metodę wyliczania współczynników szorstkości w oparciu o wzory na z_0 i z_d dla nieregularnych grup budynków opracowane przez Bottemę i Mestayera [3]. Do opracowania rozkładów przestrzennych tych współczynników badany obszar podzielili natomiast, zgodnie z podejściem Grimmonda i Oke'a [5], na obszary przypisywane do każdego obiektu w bazie budynków. Pola odniesienia wyznaczane są tak, by granice pomiędzy sąsiednimi obszarami przebiegały w geometrycznym środku odcinków łączących obiekty, do których są przypisywane.

W przyjętej metodzie, współczynnik z_0 definiuje się następująco:

$$z_0 = (h - z_d)\exp\left(\frac{-\kappa}{\sqrt{(0{,}5 C_{dh}\lambda_F)}}\right) = (h - z_d)\exp\left(-\sqrt{\frac{0{,}4}{\lambda_F}}\right)$$

gdzie:

κ – stała von Karmana wynosząca 0,4;

C_{dh} – współczynnik oporu dla odosobnionych przeszkód wynoszący 0,8.

Pozostałe wielkości występujące we wzorze:

h – objętościowo uśredniona wysokość budynków:

$$h = \frac{\sum_{i=1}^{n} V_i h_i}{\sum_{i=1}^{n} V_i}$$

λ_F – stosunek bocznej powierzchni budynków normalnej do kierunku napływu wyrażany wzorem:

$$\lambda_F(\theta) = \frac{A_F(\theta)}{A_T}$$

gdzie

$A_F(\Theta)$ – suma pól rzutów powierzchni bocznych budynków, normalnych do kierunku napływu (stąd zależność od kąta Θ);

A_T – pole powierzchni obszaru przypisanego do danego obiektu.

Współczynnik z_d – wysokość nowej płaszczyzny zerowej dla danego obszaru (przypisanego do budynku lub ścisłej grupy), definiuje się następująco:

$$z_d = h(\lambda_P)^{0{,}6}$$

gdzie λ_P:

$$\lambda_P = \frac{A_P}{A_T}$$

gdzie

A_P – pole powierzchni rzutu budynków (lub grupy) na płaszczyznę poziomą;

A_T – pole powierzchni przypisanej do danego obiektu.

Ponieważ λ_F zależy od kierunku napływu powietrza, w pierwszym etapie obliczeń, podobnie jak w przypadku prac Gala i Ungera [5], wykonano wyliczenia dla ośmiu głównych kierunków geograficznych, a następnie wyliczono średnią wartość współczynników szorstkości, przypisując ją każdemu z pojedynczych obszarów przypisanych do budynków (lub ich grup).

Algorytmy Gala, Sümeghy i Ungera przewidywały uwzględnienie zróżnicowanej wysokości w obrębie grup budynków łączących się w jedną bryłę. Takie dane nie zawsze są dostępne. Przyjęto założenie, że grupa budynków będzie charakteryzowana jedną, średnią wielkością.

5.3. Realizacja algorytmu obliczeniowego w środowisku systemu GRASS

Pierwszym krokiem prowadzonych obliczeń jest przypisanie każdemu budynkowi/kompleksowi budynków unikalnego identyfikatora. Do tego celu wykorzystano polecenie r.clump. Składnia polecenia wygląda następująco:

```
r.clump input=bud output=bud.clump
```

Warstwa rastrowa zawierająca wysokości budynków nosi nazwę „bud”. Polecenie tworzy nową warstwę o nazwie „bud.clump”, w której elementy rastra każdego budynku mają nadany unikalny identyfikator przypisany temu budynkowi.

W kolejnym kroku wygenerowano powierzchnie oddziaływania. Do tego celu wykorzystano polecenie r.grow. Polecenie to tworzy strefy buforowe wokół budynków. Polecenie to uruchomiono wewnątrz skryptu, który powiększał bufor wokół budynków w kilku krokach. Pozwoliło to znaczne przyspieszenie obliczeń. Skrypt generował warstwy dla różnych zasięgów strefy buforowej.

Treść skryptu tworzącego powierzchnie oddziaływania:

```
#!/bin/sh

warstwa=$1
d=$2
r=$3
max=$4
while [ $d -lt $max ]
   do
           echo "$d / $max"
           d1=$(($d+$r))
           r.grow input=$warstwa$d
                  output=$warstwa$d1
                  radius=$r
           d=$d1
   done
```

Pomocniczo przygotowano warstwy zawierające odległości pomiędzy elementami rastra liczone wzdłuż osi x, y oraz w kierunku przekątnych. Formuły realizujące te obliczenia wykorzystywały polecenie r.mapcalc:

```
r.mapcalc 'Y=round(100*(y()-
   232617.5))'
r.mapcalc 'X=round(100*(x()-
   553717.5))'
r.mapcalc 'XY=round((Y-X)/sqrt(2))'
r.mapcalc 'YX=round((Y+X)/sqrt(2))'
```

Po obliczeniu warstw pomocniczych wyliczono statystyki dla budynków oceniające ich rozciągłość w 8 kierunkach:

```
r.statistics base=bud.clump cover=X
   method=min output=bud.clump.Xmin
r.statistics base=bud.clump cover=X
   method=max output=bud.clump.Xmax
```

Obliczenia powtórzono tworząc warstwy:

- bud.clump.Xmin
- bud.clump.Xmax
- bud.clump.Ymin
- bud.clump.Ymax
- bud.clump.XYmin
- bud.clump.XYmax
- bud.clump.YXmin
- bud.clump.YXmax

Następnie pobrano przypisanie statystyk do identyfikatorów budynków z plików znajdujących się w podkatalogu „cats” katalogu mapsetu. Nazwy plików tekstowych odpowiadały nazwom warstw.

Kolejnym etapem było obliczenie powierzchni S budynków, jak i powierzchni oddziaływania:

```
r.stats –a input=bud.clump
   output=bud.clump.S
r.stats –a input=bud.clump.grow250
   output=bud.clump.grow.S
```

Otrzymane pliki wraz z plikiem wysokości (bud.clump.H) połączono w jedną tabelę przypisując wyliczone wielkości do identyfikatora budynku. Wykorzystano dwa programy narzędziowe: `pr` oraz `awk`.

```
pr –tm J bud.clump.H bud.clump.S
   bud.clump.grow.S bud.clump.Xmin
   bud.clump.Xmax bud.clump.Ymin
   bud.clump.Ymax bud.clump.XYmin
   bud.clump.XYmax bud.clump.YXmin
   bud.clump.YXmax | awk '{print $1"
"$2" "$4" "$6" "$8" "$10" "$12" "$14"
   "$16" "$18" "$20" "$22}' >
   bud.stats
```

Plik z tabelą statystyk posłużył do wyliczenia z_0 oraz z_d. W rezultacie otrzymano plik wynikowy zawierający identyfikator budynku (powierzchni oddziaływania) oraz wyliczone parametry.

Poniższy skrypt realizował to zadanie:

```
awk '{
   dX=($6-$5)/100;
   dY=($8-$7)/100;
   dXY=($10-$9)/100;
   dYX=($12-$11)/100;
   if((dX>0) && (dY>0) && (dXY>0) &&
   (dYX>0) && ($4>0)) {
          h=$2;
          lp=$3/$4;
          zd=h*lp^0.6;
          lfX=h*dX/$4;
          z0X=(hzd)*exp(-
   sqrt(0.4/lfX));
          lfY=h*dY/$4;
          z0Y=(h-zd)*exp(-
   sqrt(0.4/lfY));
          lfXY=h*dXY/$4;
          z0XY=(h-zd)*exp(-
   sqrt(0.4/lfXY));
          lfYX=h*dYX/$4;
          z0YX=(h-zd)*exp(-
   sqrt(0.4/lfYX));
          print $1" "zd"
   "(z0X+z0Y+z0XY+z0YX)/4;
   }
}' bud.stats >bud.zdz0
```

Otrzymany plik „bud.zdz0” wykorzystano do stworzenia reguł reklasyfikacji pliku „bud.clump”. Ponieważ reklasyfikację można wykonywać jedynie używając liczb całkowitych, wielkości otrzymanych parametrów najpierw pomnożono przez 10^6, a następnie uzyskane warstwy podzielono przez tą wartość.

```
awk '{print $1"="int($2*1000000)}'
   bud.zdz0 >bud.zd.rules
awk '{print $1"="int($3*1000000)}'
   bud.zdz0 >bud.z0.rules
r.mapcalc
   'bud.grow.zd=bud.grow.reclass.zd/10
   00000.0'
r.mapcalc
   'bud.grow.z0=bud.grow.reclass.z0/10
   00000.0'
```

W ten sposób otrzymano dwie warstwy zawierające współczynniki szorstkości o nazwach „bud.grow.z0” oraz „bud.grow.zd”.

Współczynniki szorstkości zostały wyliczone dla powierzchni oddziaływania. Pozostała powierzchnia, w szczególności na terenach podmiejskich, obszary z dala od budynków, nie określono miała określonej długości szorstkości. Obszarom tym przypisano wielkość długości szorstkości zgodnie z klasyfikacją Davenporta [4]. Do obliczeni wykorzystano warstwę „pokr” zawierającą klasyfikacje pokrycia terenu:

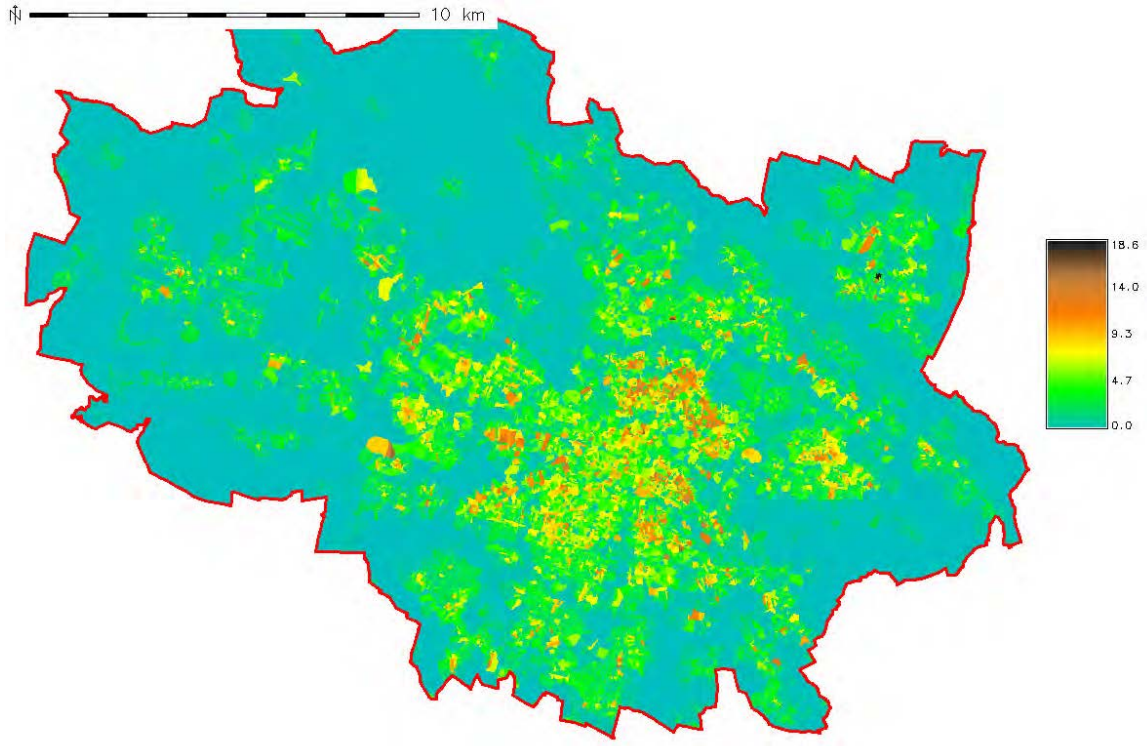
- woda (pokr=1) - z_0=0.0002
- budynki (bud_z0) - z_0=bud.z0
- zieleń średnia (pokr=3) - z_0=0.5
- zieleń wysoka (pokr=2) - z_0=1.5
- pozostałe tereny (trawa) – z_0=0.01

```
r.mapcalc 'z0=if(pokr==1,0.0002,if(!
   isnull(bud.grow.z0),bud.grow.z0,if
   (pokr==3,0.5,if(pokr==2,1.5,0.01)))
   )'
```

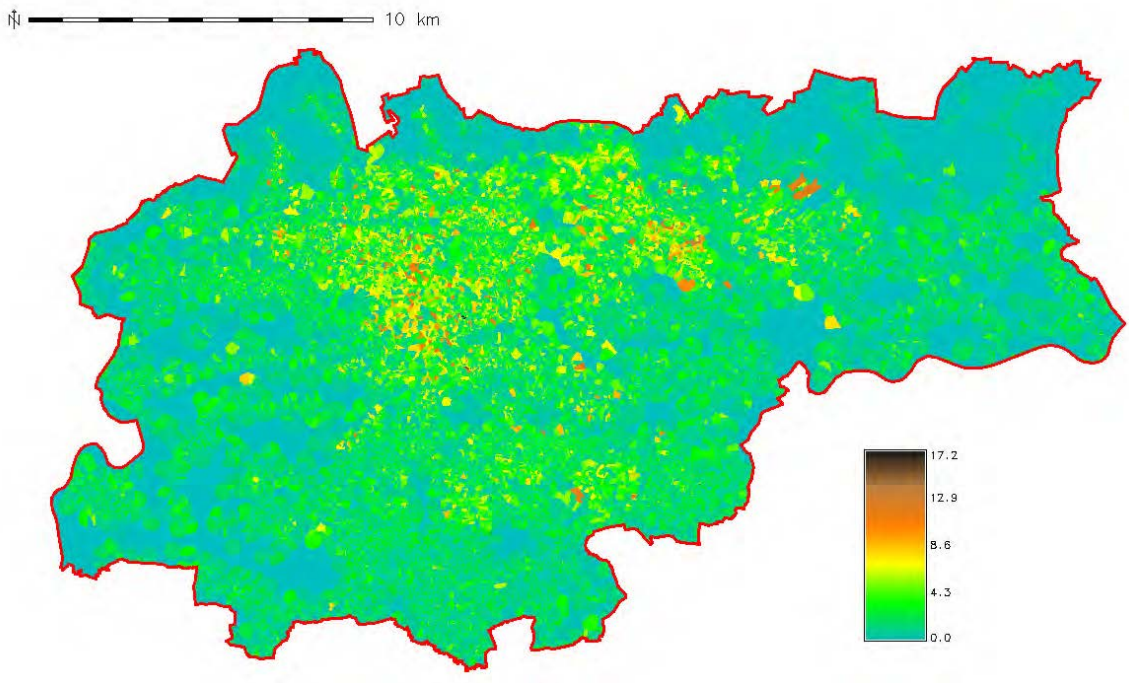
W ten sposób otrzymano warstwę rastrową zawierającą informację o długości szorstkości określoną dla całego obszaru badań.

5.4. Rozkład przestrzenny współczynników szorstkości dla Wrocławia i Krakowa

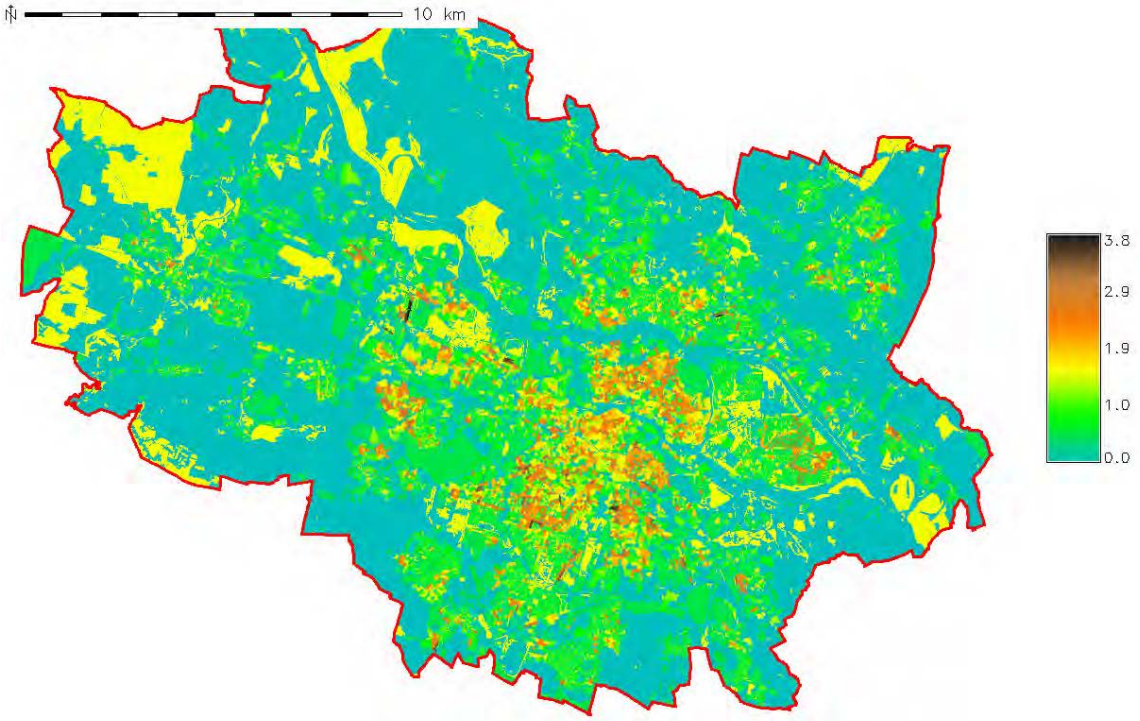
Znając wysokości wszystkich obiektów reprezentujących zabudowę na przyjętych obszarach badań, a także pozostałe cechy obiektów (pola powierzchni rzutów na płaszczyznę poziomą i płaszczyzny normalne do ośmiu głównych kierunków napływów), możliwe było wykonanie obliczeń współczynników z_0 i z_d zgodnie z algorytmami wprowadzonymi przez Gala, Sümeghy i Ungera. Wyniki obliczeń współczynnika z_d dla obiektów znajdujących się na obszarach badań we Wrocławiu i Krakowie przedstawiają rysunki 5.2 i 5.3.



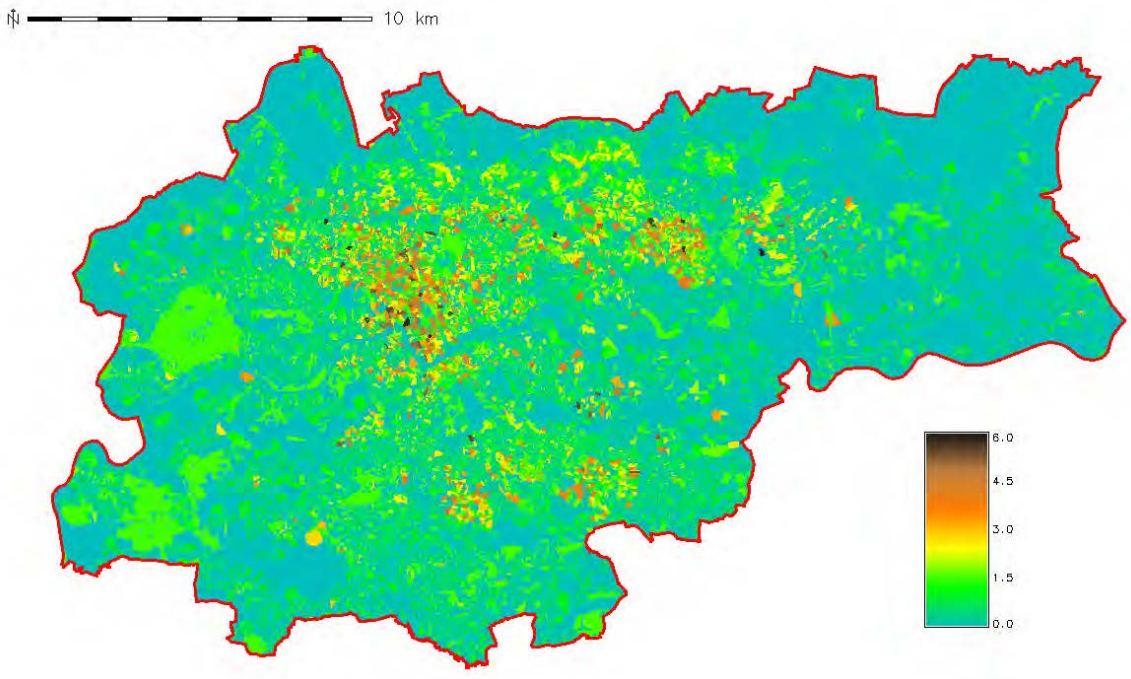
Rys. 5.2 Mapa wynikowa – rozkład przestrzenny współczynnika szorstkości z_d – Wrocław



Rys. 5.3 Mapa wynikowa – rozkład przestrzenny współczynnika szorstkości z_d – Kraków



Rys. 5.4 Mapa wynikowa – rozkład przestrzenny współczynnika szorstkości z_0 – Wrocław



Rys. 5.5 Mapa wynikowa – rozkład przestrzenny współczynnika szorstkości z_0 – Kraków

Współczynnik z_d we Wrocławiu osiąga maksymalnie wartość 18.63 m, dla Krakowa przyjmuje zaś maksymalnie wartości do 17.23 m. Przypisane do budynków obszary stanowiły strefy, których granice rozciągały się maksymalnie 250 m od obiektu (budynku, lub ich grup). W terenie miejskim, gęsto zabudowanym, gdzie obiekty znajdują się w mniejszych odległościach niż 500 m od siebie, udało się dzięki temu wykreślić ciągłą mozaikę obszarów, dla których wyliczane były współczynniki szorstkości.

Dla obszarów pokrytych wodą i roślinnością przyjęto dodatkowo wartości średnie współczynników z_0 przypisywane tym terenom, a wynoszące 0.0002 m dla wody, 0.5 m dla zieleni o średniej wysokości (krzewy, tereny ogródków działkowych), 1.5 m dla zieleni wysokiej i 0.01 m dla terenów pokrytych trawą, łąk [4, 5]. Wielkość współczynnika szorstkości przypisywaną terenom pokrytym trawą, przypisano również obszarom, których charakterystyki nie były znane. Na rysunkach 5.4 i 5.5 przedstawiono mapy rozkładu współczynnika z_0 dla terenu badań we Wrocławiu i Krakowie.

Współczynnik z_0 we Wrocławiu nie przekracza wartości 3.83 m, dla Krakowa przyjmuje zaś wartości do 5.96 m.

5.5. Podsumowanie

Metody opracowane przez Gala, Sümeghy i Ungera dla wyliczania z_0 i z_d, dają w rezultacie szczegółowe rozkłady przestrzenne współczynników szorstkości. Wyznaczenie map rozkładów dla obszarów całych miast nie stanowi już problemu. Metody te dają w efekcie dużo bardziej szczegółowy obraz tych parametrów w obszarach zurbanizowanych i w ich otoczeniu, aniżeli metody wykorzystujące jako podstawę bazy danych zawierające klasyfikację użytkowania terenu.

Współczynniki z_0 i z_d wyznaczone dla obu miast klasyfikują większość terenów zabudowanych do grupy terenów o niskiej szorstkości i w niewielkim stopniu modyfikujących pole wiatru (obszary o małej wysokości i gęstości zabudowy).

Jedynie niewielkie powierzchniowo obszary klasyfikują się do grupy o średniej i wysokiej szorstkości, w znacznym stopniu modyfikującej pole wiatru. Dodatkowo na zmniejszenie się współczynników szorstkości ma niebagatelny wpływ obszar zajmowany przez zieleń wysoką i średnią w obu miastach.

Istnieją różnice w wielkości powierzchni obszaru zajmowanego przez tereny zurbanizowane obu miast. Niemniej, tereny zabudowane zarówno we Wrocławiu i w Krakowie mają zbliżoną charakterystykę – zdecydowana większość zabudowy powoduje przesunięcie powierzchni zerowej o około 2 m, a długość szorstkości nie przekracza 1.2 m, co plasuje Wrocław i Kraków na granicy klas określonych przez Grimmonda i Oke'a [5] jako:

- miasta o niskiej zabudowie i gęstości
- miasta o średniej wysokości zabudowy i gęstości.

Rozkład przestrzenny współczynnika szorstkości z_0 wykorzystany został, jako jedna z danych wejściowych, do określenia zasięgu warstwy mieszania dla obu miast.

Wykorzystanie systemu GRASS jak również pomocniczych programów (`awk`, `pr`) pozwoliło na zaimplementowanie tej metody przy pomocy oprogramowania FOSS (ang. *Free Open Source Software*). Wydajność obliczeń w systemie GRASS jest na tyle wysoka, że obliczenia dla miast Wrocław i Kraków trwały kilkanaście minut. Obliczenia prowadzone były na warstwach rastrowych w rozdzielczości siatki 1 m x 1 m. Skrypty, które wykorzystano w trakcie obliczeń pozwalały na kontrolę poprawności obliczeń na każdym etapie.

Kolejnym krokiem w rozwoju przedstawionej implementacji algorytmu obliczania szorstkości będzie uwzględnienie w obliczeniach dokładnego modelu wysokościowego budynków odwzorowującego kształt dachów.

Literatura

[1] Arya S. P. S., 1981: *Parameterizing the height of the stable atmospheric boundary layer*, J. Appl. Meteorol. 20, s. 1192-1202

[2] Baklanov A. et al., 2006: *Towards estimating the mixing height in urban areas*, Scientific Report 06-06, Ministry of Transport and Energy, Copenhagen, ss. 41

[3] Bottema M., Mestayer P.G. (1998), *Urban roughness mapping - validation techniques and some first results*, Journal of Wind Engineering and Industrial Aerodynamics, 1998; 74–76, s. 163–173
[4] Davenport A., Grimmond S., Oke T. , Wieringa J. (2000), *The revised Davenport roughness classification for cities and sheltered country*, 3rd Symp. On the Urban Environment, 14-18.08.2000, Davis, California, AMS Proceedings, s. 9-10
[5] Grimmond C. S. B., Oke T. R. (1999), *Aerodynamic Properties of Urban Areas Derived from Analysis of Surface Form*, Journal of Applied Meteorology, Vol. 38, September 1999, s. 1262-1292
[5] Gál T., Sümeghy Z. (2007), *Mapping The Roughness Parameters In a Large Urban Area For Urban Climate Applications*, Acta Climatologica Et Chorologica, Universitatis Szegediensis, Tomus 40-41, 2007, s. 27-36
[7] Gal T., Unger J. (2009), *Detection of ventilation paths using high-resolution roughness parameter mapping in a large urban area*, Building and Environment, 44 (2009), s. 198-206
[8] Garratt J.R., 1992, *The Atmospheric Boundary Layer,* Cambridge University Press, ss. 316
[9] Nieuwstadt F. T. M., 1981: The steady state height and resistance laws of the nocturnal boundary layer: theory compared with Cabauw observations. Boundary-Layer Meteorol. 20, s. 3-17
[10] Venkatram A., 1980: Estimating the Monin-Obukhov length in the stable boundary layer for dispersion calculations, Boundary-Layer Meteorol., 18, 481
[11] Zilitinkevich S., 1972: On the determination of the height of the Ekman boundary layer. Boundary-Layer Meteorol., 3, s. 141-145
[12] Zilitinkevich S., Mironov D., 1996: A multi-limit formulation for the equilibrium depth of a stably stratified boundary layer. Boundary-Layer Meteorology, 81(3-4), s. 325-351
[13] Zilitinkevich S., Baklanov A., 2002: Calculation of the height of stable boundary layers in practical applications. Boundary-Layer Meteorol., 105(3), s. 389-409
[14] Zilitinkevich S., Baklanov A., Rost J., Smedman A.-S., Lykosov V., Calanca P., 2002: Diagnostic and prognostic equations for the depth of the stably stratified Ekman boundary layer. Quart. J. Roy. Meteorol. Soc., 128, s. 25-46

6. Modelowanie wysokości pokrywy śnieżnej w Sudetach Zachodnich

Hanna Ojrzyńska

6.1. Wstęp

Prężny rozwój oprogramowania Open Source GIS, szczególnie popularnego w środowiskach akademickich, daje nowe możliwości wielowymiarowych analiz przestrzennych. Stosowany w polskiej klimatologii system GRASS (Geographic Resources Analysis Support System) stanowi zbiór odrębnych programów uruchamianych bezpośrednio z poziomu systemu operacyjnego [8, 9, 10], dzięki czemu możliwa jest automatyzacja poleceń zebranych w skrypty, bądź współpraca z dodatkowym oprogramowaniem (np. pakietem statystycznym R). Fakt ten pozwala na badanie zależności między dużą liczbą czynników zmiennych w czasie i przestrzeni, co szczególnie istotne jest podczas opisu rzeczywistego i modelowanego rozkładu wysokości pokrywy śnieżnej.

Czynnikami mającymi bezpośredni wpływ na miąższość śniegu są elementy meteorologiczne: suma i rodzaj opadu, temperatura, wilgotność powietrza, prędkość i kierunek wiatru czy natężenie promieniowania słonecznego [1, 14, 18]. Wartości wymienionych elementów są silnie uwarunkowane czynnikami hipsometrycznymi i morfologicznymi oraz rodzajem pokrycia terenu, stąd zainteresowanie klimatologów numerycznymi modelami terenu. Zestawienie, generowanych z DEM (Digital Elevation Model udostępniony przez NASA; www.jpl.nasa.gov/srtm) i CORINE (Coordination of Information on the Environment Land Cover; www.eea.europa.eu/themes/landuse/clc-download), warstw rastrowych - wskaźników opisujących rzeźbę i rodzaj pokrycia terenu z wielkościami wysokości pokrywy śnieżnej umożliwia w sposób pośredni określenie wpływu meteorologicznych determinant rozkładu miąższości pokrywy. W efekcie podstawowy model terenu wykorzystywany jest jako podstawa modelowania klimatologicznego [3, 12].

Celem niniejszej pracy jest uwypuklenie roli systemu GRASS i pakietu statystycznego R w określaniu wpływu poszczególnych uwarunkowań i modelowaniu zjawisk meteorologicznych na przykładzie wysokości pokrywy śnieżnej. Szczegółowy opis dotyczyć będzie zastosowanej metodyki i kolejności działań. Wyniki modelowania opublikowane zostały już w 2010 [11], dlatego w obecnym opracowaniu potraktowane zostaną marginalnie.

6.2. Zbiór zmiennych zależnych i niezależnych

Zasadniczym krokiem poprzedzającym opis przestrzennego zróżnicowania wysokości pokrywy śnieżnej w Sudetach Zachodnich było przygotowanie warstw wektorowych zawierających wyniki rzeczywistych pomiarów wysokości pokrywy śnieżnej z sezonu zimowego 2003/2004 w terenie badań (wyniki obserwacji ze stacji i posterunków IMGW i CHMU oraz pomiarów patrolowych; łącznie 160 punktów). Z uwagi na znaczną zmienność wysokości pokrywy śnieżnej w czasie sezonu zimowego zdecydowano o opracowaniu osobnych zestawień dla „dnia charakterystycznego” w fazach: wzrostu wysokości pokrywy, maksymalnej wysokości pokrywy i zaniku pokrywy. Niezależnie od siebie zestawiono wyniki pomiarów z punktów leżących w obrębie lasu oraz punktów zlokalizowanych poza nim [11]. Zestawienia zapisane w postaci plików .csv zawierały kolejno kolumny ze współrzędnymi punktu

pomiarowego (X i Y w układzie PUWG 92), wartość wysokości pokrywy śnieżnej i numer porządkowy (kategorię) punktu. Import do warstwy wektorowej GRASS dokonany został przy pomocy komendy:

```
v.in.ascii
   input=ścieżka_dostępu/plik.csv
   output=nowa_nazwa format=point
   fs=; skip=0 x=1 y=2 z=0 cat=4
```

gdzie wskazano kolejno:

- nazwę pliku wejściowego wraz ze ścieżką dostępu,
- nazwę wyjściowej warstwy wektorowej,
- typ geometryczny importowanych danych,
- znak podziału pól w pliku wejściowym,
- numer wiersza z nagłówkami z pliku wejściowego do pominięcia podczas importu (0=brak), kolejne numery kolumn w pliku wejściowym zawierające współrzędną X, Y i numer kategorii.

W efekcie otrzymano zbiór 6 warstw wektorowych zawierających informację o miąższości śniegu w danej fazie rozwoju pokrywy. Z uwagi na przyjęty cel analiz zmienna ta będzie dalej nazywana zmienną zależną.

Opracowanie zbioru zmiennych niezależnych wiązało się z wyszczególnieniem czynników związanych z rzeźbą terenu, które różnicują pola elementów meteorologicznych istotnych w procesie akumulacji i zaniku pokrywy śnieżnej. Czynniki te, nazywane tu wskaźnikami morfometrycznymi, reprezentują liczny zbiór parametrów opisujących wysokość formy, jej kształt, stopień rozczłonkowania czy ekspozycję. Każdy ze wskaźników stanowi osobną warstwę rastrową przygotowaną w oparciu o hipsometryczny model terenu DEM o rozdzielczości 50 m x 50 m. Bezpośrednia informacja o wysokości nad poziom morza ze wspomnianego modelu posłużyła za pośredni wskaźnik częstości opadów śniegu. Wzrost częstości opadów wraz ze wzrostem wysokości nad poziom morza związany jest z pionowym spadkiem temperatury powietrza i prężności pary wodnej w atmosferze.

Zwartość bariery górskiej oraz wyeksponowanie stoków masywu staje się istotne w zestawieniu z przeważającym kierunkiem napływu wilgotnych mas powietrza, warunkując wyższe sumy opadów na stokach dowietrznych, przy cieniu opadowym na stokach zawietrznych. W Sudetach, ze względu na występowanie efektów transfluencyjnych [7], strefa największych opadów przesunięta jest na stronę zawietrzną. W konkretnych sytuacjach barycznych przebieg masywu prostopadły do kierunku adwekcji oraz jego znaczna zwartość powoduje powstanie efektów fenowych odpowiedzialnych za szybki zanik pokrywy śnieżnej na stokach objętych spływem ogrzanego adiabatycznie, względnie suchego powietrza [6].

Wygenerowanie warstw rastrowych opisujących zwartość masywu oraz jego ekspozycję wobec dominującego kierunku napływu mas powietrza (wskaźnik zasłonięcia i wyeksponowania formy) związane było z zastosowaniem analizy sąsiedztwa.

Wskaźnik zwartości obliczono jako średnią arytmetyczną wysokości gridów sąsiednich z warstwy DEM przy pomocy komendy:

```
r.neighbors input=DEM output=zwartosc
   size=3
```

gdzie wskazano kolejno

- nazwę warstwy wejściowej,
- nazwę warstwy wyjściowej,
- rozmiar pola sąsiedztwa (pole sąsiedztwa stanowi kwadrat o boku opisanym liczbą gridów z centralnie położonym gridem, dla którego liczona jest statystyka; tu rozmiar równy 3 gridy oznacza że grid centralny otoczony jest z każdej strony 1 gridem o rozdzielczości wejściowej DEM = 50 m).

Ograniczenia związane z kształtem pola sąsiedztwa (tylko kwadrat) oraz wagom składających się na nie gridów (waga 1 dla wszystkich gridów w obrębie pola sąsiedztwa)

sprawiły, że do konstrukcji wskaźnika zasłonięcia i wyeksponowania formy zastosowano analizę sąsiedztwa przy użyciu komendy:

```
r.mfilter input=DEM
   output=zwartosc_eksponowana
   filter=ścieżka/nazwa_filtra
   repeat=1
```

gdzie wskazano kolejno

- nazwę warstwy wejściowej,
- nazwę warstwy wyjściowej,
- ścieżkę dostępu i nazwę pliku filtra,
- liczbę powtórzeń wykonywanej operacji.

Plik filtru przygotowany jako plik .txt przy pomocy układu cyfr „0” i „1” definiował rozmiar i kształt pola sąsiedztwa oraz wagi poszczególnych gridów. Poniżej przedstawiono wzór zapisu filtra dla sektora kierunkowego NW - wycinka koła o promieniu 4 gridów (200 m wg opisywanego DEM):

```
TITLE
MATRIX 7
0 0 0 1 0 0 0
0 0 1 1 0 0 0
0 1 1 1 0 0 0
1 1 1 1 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
DIVISOR 10
TYPE P
```

W celu określenia stopnia zasłonięcia lub wyeksponowania danego grida z określonego sektora kierunkowego zastosowano szereg filtrów dla 4 głównych sektorów (NW, SW, NE, SE) oraz odległości sąsiedztwa od 500 m do 50 km (0.5 km, 1 km, 2 km, 5 km, 7.5 km, 10 km, 15 km, 20 km, 25 km, 30 km, 50 km). Otrzymane warstwy zawierały informację o średniej wysokości w zadanym sektorze kierunkowym. W ostatnim kroku warstwę tą odjęto od warstwy DEM przy pomocy kalkulatora warstw rastrowych, jak w przykładzie:

```
r.mapcalc 'zasloniecie10NW=DEM-
   zwartosc_eksponowana10NW'
```

gdzie wskazano kolejno

- nazwę warstwy wynikowej,
- przeprowadzone działanie odejmowania, gdzie odjemną jest warstwa DEM, a odjemnikiem warstwa średniej wysokości w sektorze kierunkowym NW o promieniu 10km.

Ujemne wartości wskaźnika zasłonięcia i wyeksponowania wskazują na fakt zasłonięcia danej formy terenu z rozpatrywanego sektora kierunkowego, wartości dodatnie są zaś dowodem na wyeksponowanie formy (rys. 6.1).

Analiza sąsiedztwa i działania na kalkulatorze map rastrowych były podstawą konstrukcji wskaźnika wklęsłości i wypukłości formy terenu. Podobnie jak w przypadku wskaźnika zasłonięcia i wyeksponowania, od aktualnej wysokości gridów w warstwie DEM odjęto średnią wysokość z sąsiedztwa (wskaźniki zwartości terenu) rozpatrywanego dla różnych odległości - od 500 m do 50 km, według formuły:

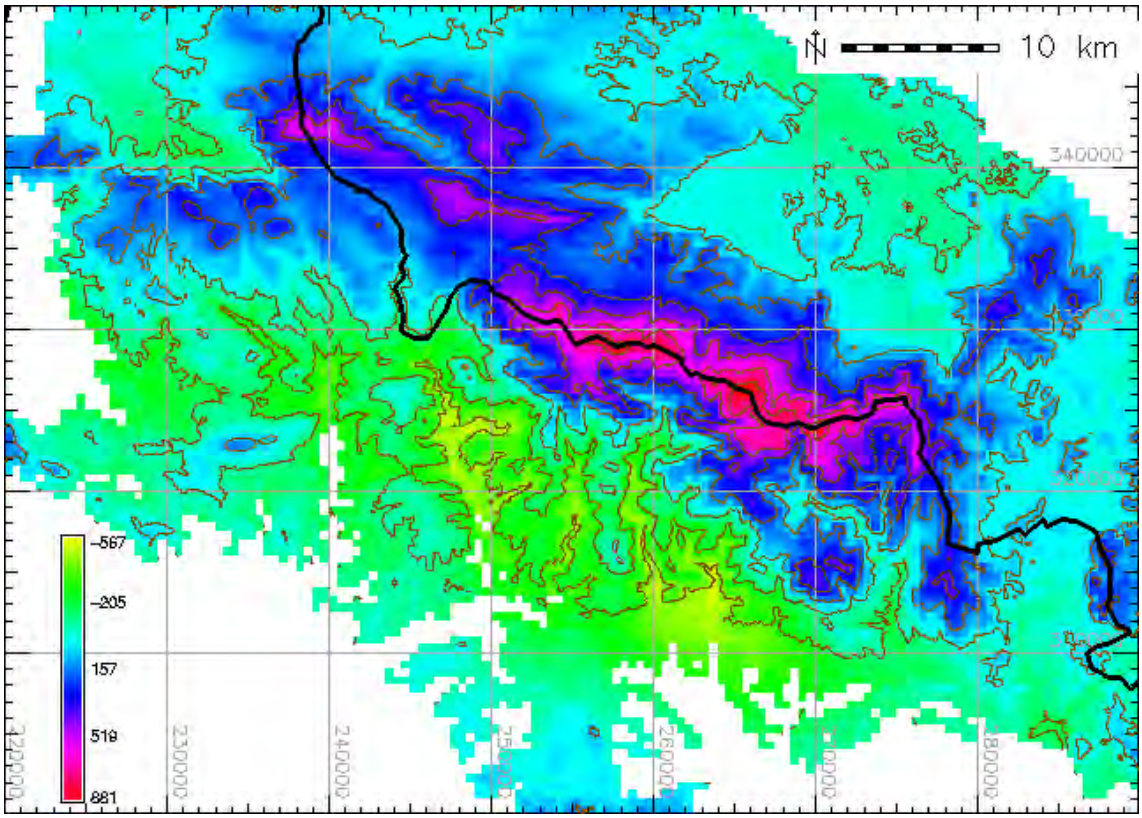
```
r.mapcalc 'forma=DEM-zwartosc'
```

gdzie wskazano kolejno

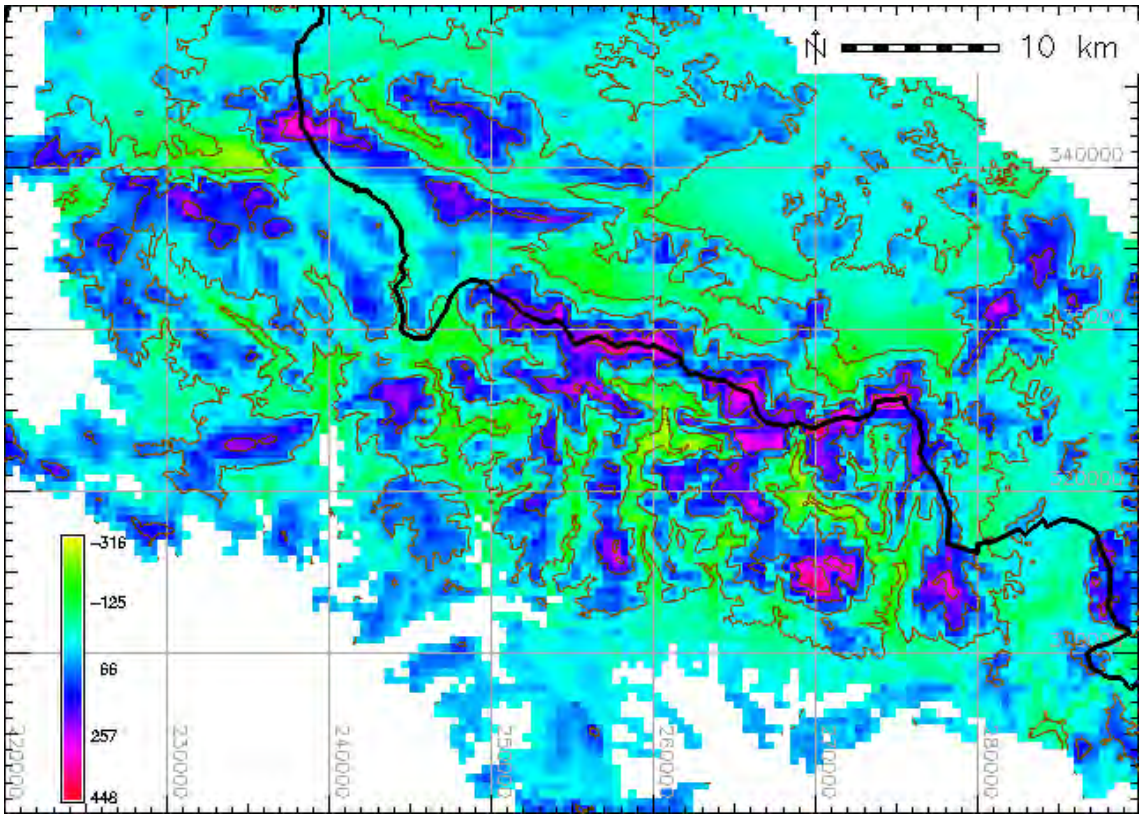
- nazwę warstwy wynikowej,
- przeprowadzone działanie odejmowania, gdzie odjemną jest warstwa DEM, a odjemnikiem warstwa średniej wysokości z sąsiedztwa.

Dodatnie wartości wskaźnika wskazują na wypukłość formy wobec otoczenia, natomiast ujemne na jej wklęsłość (rys. 6.2). Rozróżnienie typu formy terenu ma w przypadku kształtowania wysokości pokrywy śnieżnej szereg znaczeń. Forma wypukła stanowi miejsce swobodnego przewiewania zgromadzonego śniegu, który z kolei najczęściej akumulowany jest w zagłębieniach terenowych, z uwagi na wyhamowanie prędkości wiatru na ścianach zagłębień (skokowa zmiana szorstkości podłoża).

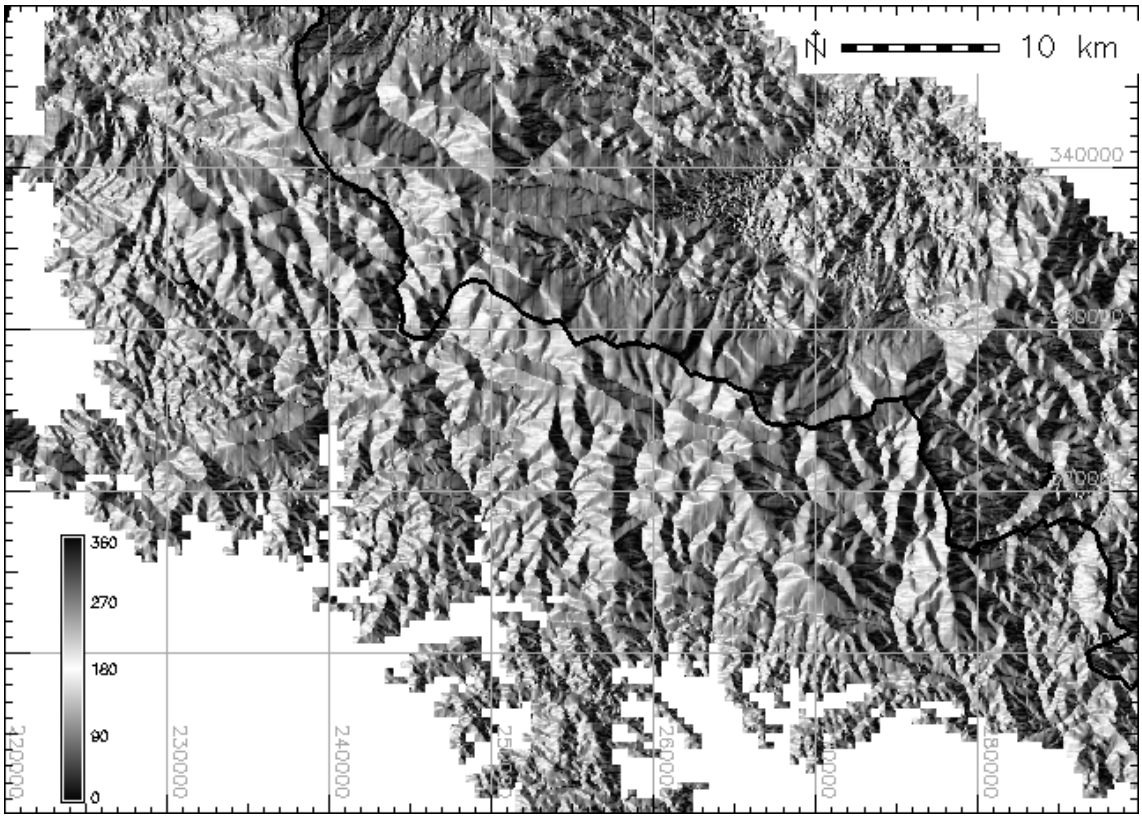
W sezonie zimowym formy wklęsłe objęte są najczęściej zastoiskami zimnego powietrza, co w połączeniu z częstym zacieniem dna zagłębienia stanowi doskonałe warunki do konserwacji zakumulowanej pokrywy. W przeciwieństwie do nich, formy wypukłe narażone są na silną, bezpośrednią operację słoneczną sprzyjającą, szczególnie w okresie wiosennym, szybkiemu zanikowi pokrywy.



Rys. 6.1 Wskaźnik zasłonięcia i wyeksponowania z sektora NE o promieniu 15km dla terenu Sudetów Zachodnich.



Rys. 6.2 Wskaźnik wklęsłości i wypukłości formy w Sudetach Zachodnich dla sąsiedztwa 15km.



Rys. 6.3 Ekspozycja formy w obrębie Sudetów Zachodnich.

W obrębie poszczególnych wypukłych, jak i wklęsłych form terenu zróżnicowanie wysokości pokrywy śnieżnej warunkowane jest także ilością promieniowania docierającego do powierzchni śniegu, a w szczególności promieniowania przez nią pochłoniętego.

Za zjawisko to odpowiada w dużej mierze odmienna ekspozycja stoków, jak i ich nachylenie [13]. Uzyskanie warstw zawierających te wskaźniki morfometryczne możliwe było przy zastosowaniu komendy r.slope.aspect w oparciu o model DEM:

```
r.slope.aspect elevation=DEM
   slope=nachylenie aspect=ekspozycja
```

gdzie wskazano kolejno

- nazwę wejściowej warstwy hipsometrycznego modelu terenu,
- nazwę warstwy wyjściowej nachylenia,
- nazwę warstwy wyjściowej ekspozycji terenu.

W wyniku opisanych działań wygenerowano ostatecznie 62 warstwy rastrowe wchodzące w skład zbioru wskaźników morfometrycznych. Dalsza, statystyczna analiza ich wpływu na różnicowanie wysokości pokrywy śnieżnej, wymagała wyodrębnienia wielkości poszczególnych wskaźników dla punktów pomiaru miąższości śniegu. Operację tą wykonano przy użyciu komendy r.what i wspominanego wcześniej pliku z wynikami pomiaru wysokości pokrywy (plik.csv):

```
r.what
   input=DEM,nachylenie,ekspozycja,
   zwartosc500m,zasloniecie500mNW,
   zasloniecie500mNE,
   zasloniecie500mSW,
   zasloniecie500mSE,forma500m,
   zwartosc1km,zasloniecie1kmNW,
   zasloniecie1kmNE,zasloniecie1kmSW,
   zasloniecie1kmSE,forma1km
   (...),zwartosc50km,
   zasloniecie50kmNW,
   zasloniecie50kmNE,
   zasloniecie50kmSW,
   zasloniecie50kmSE,forma50km
   <plik.csv >wynik.txt
```

gdzie wskazano kolejno

- nazwy warstw, z których pobierana będzie wielkość wskaźnika,
- nazwę pliku zawierającego współrzędne punktów (przekierowanie zawartości pliku za pomocą znaku '<'),
- nazwę pliku wyjściowego z zapisanymi danymi (przekierowanie wyników pracy programu do pliku za pomocą znaku '>').

Tak przygotowany plik tekstowy zawierał ostateczny zbiór wielkości zmiennej zależnej i zmiennych niezależnych, który poddano analizie regresji.

6.3. Model regresji wieloczynnikowej

Model regresji wieloczynnikowej opisany jest równaniem:

$$Y = B_0 + B_1 \cdot X_1 + \ldots + B_k \cdot X_k$$

gdzie

Y - wartość estymowana (zmienna zależna),

X_i - zmienne niezależne dla $i = 1 \ldots k$,

B_i - współczynniki regresji dla $i = 1 \ldots k$.

W celu wyselekcjonowania 4 zmiennych niezależnych, mających najistotniejszy wpływ na różnicowanie wysokości pokrywy śnieżnej, posłużono się metodą regresji krokowej postępującej przy użyciu pakietu statystycznego R. Poniżej przedstawiono sekwencję najważniejszych komend użytych w trakcie analiz:

- wczytanie do tabeli data pliku wynik.txt; pozostałe parametry funkcji zawierają informacje dotyczące nagłówków, separatora tekstu i separatora dziesiętnego

```
data=read.table(file="wyniki.txt",
   header=F,sep="|",dec=".")
```

- wybranie kolumny tabeli zawierającej wartości zmiennej zależnej

```
zalezne=data[ ,c(3)]
```

- wybranie kolumn tabeli (różnych od 3 i 4) zawierających wartości zmiennych niezależnych

```
niezależne=data[ ,-c(3,4)]
```

- wskazanie zbioru zmiennych zależnych i niezależnych oraz metody krokowej postępującej dla analizy regresji wieloczynnikowej

```
m0.subs=regsubsets(x=niezalezne,
   y=zalezne,method=c(„forward"))
```

- wyświetlenie wyników dobierania modelu regresji

```
summary(m0.subs)
```

- wygenerowanie modelu liniowego (równanie estymacji pokrywy śnieżnej); Indeksy przy V wskazują numery kolumn w tabeli – określają rodzaj wskaźnika morfometrycznego

```
m0=lm(zalezne~V1+V8+V9+V23,
   niezalezne)
```

- badanie wzajemnej współliniowości elementów równania m0

```
vif(m0)
```

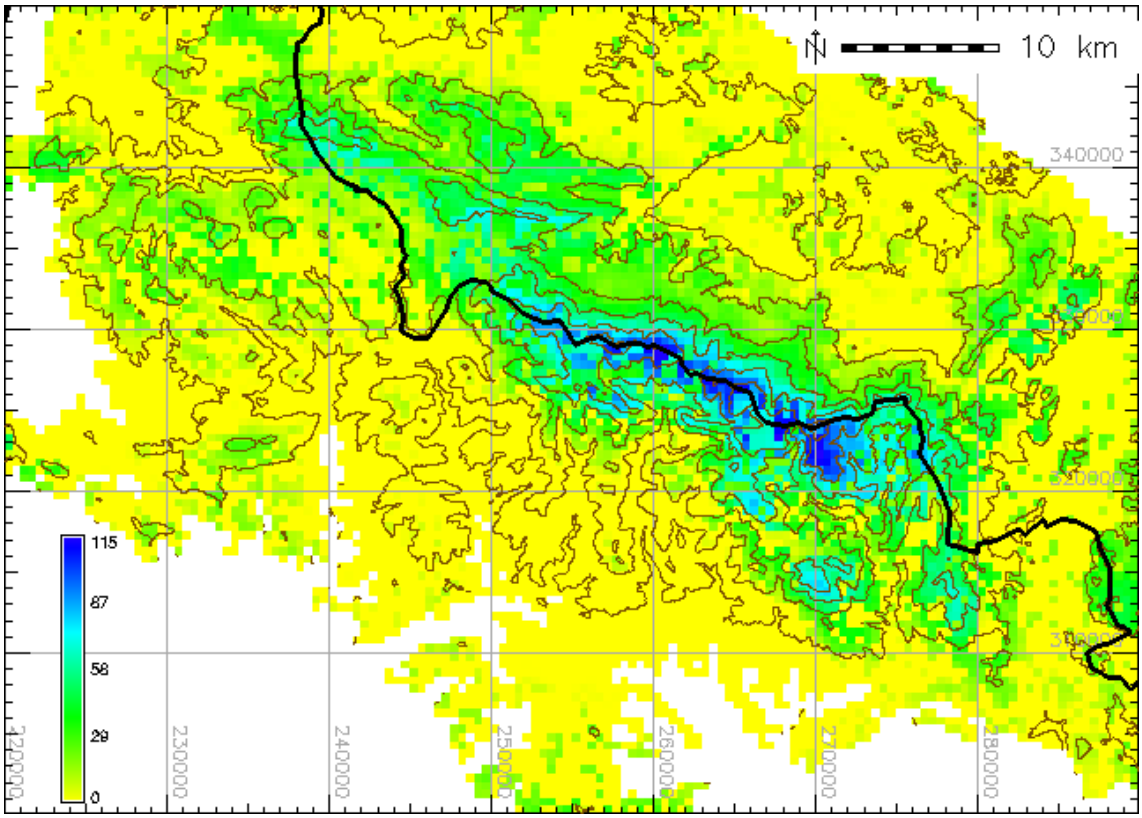
Polecenie summary(m0) dla modeli liniowych jest źródłem informacji o istotności zmiennych niezależnych (według współczynnika α), rozkładzie i błędach standardowych reszt, współczynnikach korelacji równania oraz wielkości współczynników regresji dla poszczególnych zmiennych niezależnych. Te ostatnie, po ostatecznej akceptacji wyników statystyki dla kolejnych modeli, posłużyły do konstrukcji równań estymacji wysokości pokrywy śnieżnej, które to równania następnie wykorzystano w systemie GRASS do stworzenia rozkładów przestrzennych.

6.4. Modelowanie z uwzględnieniem rodzaju pokrycia terenu

Na początku opracowania zwrócono uwagę na podział wyników pomiarów wysokości pokrywy śnieżnej na punkty leżące w obrębie lasu i poza nim. Fakt ten, opisany szerzej we wcześniejszej publikacji [11], związany był w największym stopniu z trudnościami wynikającymi z przeniesienia jakościowej informacji o rodzaju pokrycia terenu na informację ilościową (np. o szorstkości terenu czy wskaźniku widoczności nieba).

Sporządzając osobne modele dla terenów leśnych i nieleśnych skorzystano z możliwości przypisania różnych równań do odpowiednich rastrów cyfrowego modelu pokrycia terenu.

Konstrukcja modeli estymacji wysokości pokrywy opierała się na wynikach analizy regresji uzyskanych z pakietu statystycznego R.



Rys. 6.4 Przestrzenny rozkład modelowanej wysokości pokrywy śnieżnej w Sudetach Zachodnich w trzeciej fazie jej rozwoju (etap wiosennych nawrotów pokrywy).

Przy pomocy kalkulatora warstw rastrowych zestawiono współczynniki regresji wraz z odpowiadającymi im zmiennymi niezależnymi – rozkładami wskaźników morfometrycznych, jak w przykładach:

```
r.mapcalc
   'faza3las=B0+B1*zasloniecie5kmSW+
   B2*zasloniecie15kmNW+
   B3*zasloniecie15kmNE'
r.mapcalc 'faza3nielesne=B0+B1*
   zasloniecie15kmNE'
```

gdzie wskazano kolejno

- nazwy wynikowych warstw wysokości pokrywy śnieżnej w trzeciej wydzielonej fazie rozwoju pokrywy dla terenu lasu (równanie 1) i terenów nieleśnych (równanie 2),
- nazwy warstw – wskaźników morfometrycznych wraz z ich współczynnikami regresji.

Kalkulator warstw rastrowych wykorzystany został także ostatecznie podczas przypisywania konkretnego wyniku estymacji wysokości pokrywy do rodzaju pokrycia terenu, jak w przykładzie (rys. 6.4):

```
r.mapcalc
   'faza3=if(pokrycie==1,faza3las,
   faza3nielesne)'
```

gdzie wskazano kolejno

- nazwę wynikowej warstwy wysokości pokrywy śnieżnej w trzeciej wydzielonej fazie rozwoju pokrywy
- warunek: „jeśli grid w warstwie pokrycie przyjmuje wartość 1 (las+kosodrzewina) przypisz mu wartość z warstwy faza3las, a jeśli inną (tereny nieleśne) to przypisz mu wartość z warstwy faza3nielesne.

6.5. Ocena wyników

Do ostatecznej konstrukcji modeli wykorzystano tylko istotne zmienne (współczynnik $\alpha \leqslant 0.05$): wskaźniki zasłonięcia/wyeksponowania danego miesjca na adwekcję mas powietrza z sektora SW, NW i NE oraz wskaźnik formy terenu i położenia opisany współrzędną X.

Tabela 6.1 Charakterystyka statystyczna modeli regresji wieloczynnikowych dla poszczególnych faz kształtowania się pokrywy śnieżnej i dwóch kategorii pokrycia terenu

ZMIENNE		Faza wzrostu pokrywy		Faza maksymalnego ekwiwalentu wodnego		Faza wiosennych nawrotów pokrywy	
		tereny leśne	tereny nieleśne	tereny leśne	tereny nieleśne	tereny leśne	tereny nieleśne
Zasłonięcie/ wyeksponowanie w sektorze SW	50km		***				
	20km	***			***		
	10km			***			
	5km					**	
Zasłonięcie/ wyeksponowanie w sektorze NW	15km	***				*	
Zasłonięcie/ wyeksponowanie w sektorze NE	50km				**		
	15km					***	***
wskaźnik wklęsłości/ wypukłości (r=25km)			*		***		
współrzędna x				***			
R^2		0,66	0,61	0,76	0,63	0,42	0,47
MAE		12cm	10cm	27cm	28cm	34cm	32cm

(‘***’ $0<\alpha<0{,}001$; ‘**’ $0{,}001<\alpha<0{,}01$; ‘*’ $0{,}01<\alpha<0{,}05$)

Podstawowe statystyki uzyskane z polecenia summary() pakietu R dla modelu liniowego:

- współczynniki korelacji,
- wartości błędów standardowych reszt

okazały się zadowalające jedynie w przypadku fazy maksymalnej wysokości pokrywy śnieżnej (tab. 6.1). W pozostałych przypadkach, a w szczególności w fazie zaniku pokrywy śnieżnej, równanie modelu nie opisywało w pełni rzeczywistego rozkładu zjawiska.

W celu prześledzenia przestrzennego rozkładu błędów zastosowano walidację krzyżową (ang. *cross-validation*). Podstawowe wyniki opisane przez średni błąd bezwzględny (MAE - ang. *Mean Absolute Error*) (tab. 6.1) potwierdziły, że największą wartość w opisie rzeczywistego rozkładu posiadają modele dla fazy maksymalnej wysokości pokrywy. Jest to widoczne szczególnie dla obszarów leśnych. We wspomnianej fazie rozwoju pokrywy śnieżnej na obszarach pozbawionych pokrywy roślinnej, zwłaszcza w narażonej na przewiewanie śniegu wierzchowinowej partii Karkonoszy, odnotowano najwyższe wartości błędów modelu, sięgające do 40cm a w pojedynczych przypadkach nawet 60cm. W tym samym czasie w obrębie lasu wartości błędów rzadko przekraczały 35cm. Stosunkowo niewielkie wartości błędów dla modeli konstruowanych, w szczególności dla fazy wzrostu wysokości pokrywy oraz w mniejszym stopniu dla fazy jej zaniku, wynikają ze zdecydowanie niższych rzeczywistych wartości miąższości śniegu. Oba modele wymagają jeszcze dodatkowych analiz z uwzględnieniem nowych zmiennych opisujących przestrzenny rozkład elementów meteorologicznych.

6.6. Podsumowanie

Przedstawiona sekwencja działań w oparciu o system GRASS i pakiet statystyczny R stanowi przykład wykorzystania tych narzędzi w analizach meteorologicznych.

Zastosowane komendy należą do standardowych możliwości systemów opisywanych szczegółowo w dostępnych w sieci manualach oraz skryptach [8, 9, 10, 19], zagadnienia metodyczne nawiązują zaś do rozwiązań prezentowanych w literaturze

polskiej [4, 5, 17] i zagranicznej [2, 15, 16]. Z uwagi na znaczne zróżnicowanie warunków meteorologicznych w poszczególnych sezonach śnieżnych, modele, przygotowane w oparciu o jeden sezon, nie mogą być odnoszone do pozostałych. Sposób konstrukcji modelu, przedstawiony powyżej jest jednak uniwersalny, w przyszłości należy więc zastosować go w oparciu o dane wieloletnie.

Literatura

[1] Barry K.G., 1992, *Mountain weather and climate*, Routledge, London - New York, ss. 512

[2] Chapman L., Thornes J.,E., 2003, *The use of geographical information system in climatology and meteorology*, Progress in Physical Geography, Vol. 27, No. 3, s. 313-330

[3] Coughlan J.C., Running S.W. 1997. *Regional ecosystem simulation: A general model for simulating snow accumulation and melt in mountainous terrain,* Landscape Ecology 12, s. 119-136

[4] Kryza M., Sobik M., 2004, *Regresyjny model rozkładu przestrzennego temperatury powietrza na Dolnym Śląsku*, Czasopismo Geograficzne, t. 75, z. 3, s. 183-194

[5] Kryza M., Szymanowski M., Wieczorek M., 2007, *Wybrane metody interpolacji w modelowaniu map ekstremalnej temperatury powietrza (na przykładzie południowo-zachodniej Polski)*, Przegl. Geofiz., Rocznik LII, z. 1, s. 61-82

[6] Kwiatkowski J., 1979, *Zjawiska fenowe w Sudetach i na przedpolu Sudetów*, Problemy Zagospodarowania Ziem Górskich, z.20, s.243-277

[7] Kwiatkowski J., 1984, *Związki opadów atmosferycznych w Polskich Sudetach i na ich Przedpolu z czynnikami cyrkulacyjnymi*, Zakład Narodowy im. Ossolińskich, ss. 137

[8] Neteler M., Mitasova H., 2004, *Open source GIS: A GRASS GIS approach*, Kluwer Academic Publishers, Boston, ss. 417

[9] Neteler M., 2005, *GRASS 6 in a nutshell*. Open Source Geospatial '05 Conf. Univ. of Minnesota, USA

[10] Netzel P., 2004, GIS-GRASS. *Wprowadzenie do systemu*, [w:] Warsztaty Naukowe „Analizy Przestrzenne w programie GRASS", Pracownia Metod Modelowania Przestrzennego Środowiska Geograficznego Uniwersytetu Wrocławskiego, Wrocław, s. 59-102

[11] Ojrzyńska H., Błaś M., Kryza M., Sobik M., Urban G., 2010, *Znaczenie lasu oraz morfologii terenu w rozwoju pokrywy śnieżnej w Sudetach Zachodnich na przykładzie sezonu zimowego 2003/2004*, Sylwan 154(6), s. 412-428

[12] Sobik M., Urban G., Błaś M., Kryza M., Tomczyński K., 2009. *Uwarunkowania pokrywy śnieżnej w Sudetach Zachodnich sezonach zimowych 2001/2002-2005/2006*, Wiadomości Meteorologii Hydrologii Gospodarki Wodnej (w druku)

[13] Styszyńska A., 1995, *Dopływ promieniowania całkowitego Słońca do powierzchni o dowolnym nachyleniu i ekspozycji*, Prace Naukowe WSM w Gdyni, ss. 160

[14] Trepińska J., 2002, *Górskie klimaty*, Wyd. Instytutu Geografii UJ, Kraków, ss. 202

[15] Tveito O.E., Schöner W., 2002, *Applications of spatial interpolation of climatological and meteorological elements by the use of geographical information systems (GIS)*, KLIMA Report 28/02, ss. 44

[16] Tveito E.O., 2006, *Spatialisation of climatological and meteorological information by the support of GIS*, [w:] Final Report no.5 in the framework of the climatological projects in the application area of ECSN, Oslo

[17] Ustrnul Z., 2004, *Metody analizy przestrzennej w badaniach klimatologicznych*, [w:] Zastosowanie wybranych metod statystycznych w klimatologii, red. Bokwa A., Ustrnul Z., s. 65-8.

[18] Yoshino M., 1975, *Climate in a small area. An introduction to Local Meteorology,* University of Tokyo Press, ss. 549

Źródłą internetowe

[19] R Development Team 2006. R: A language and environment for statistical computing. R Foundation for Statistical Computing, http://www.r-project.org

7. Wykorzystanie GRASS w modelowaniu hydrologicznym

Robert Szczepanek

7.1. Wstęp

Wraz z upowszechnieniem systemów informacji przestrzennej podjęto próby wykorzystania tych nowych narzędzi w modelowaniu hydrologicznym. Jednym z pierwszych poważnych systemów, który dawał taką możliwość, był GRASS (Geographic Resources Analysis Support System) [9].

Rozwój matematycznych modeli hydrologicznych zdeterminowany był od zawsze aktualną wiedzą i dostępnym aparatem matematycznym. Upowszechnienie komputerów, a w szczególności zaawansowanych narzędzi do modelowania przestrzennego, otwarło nowe pola rozwoju modelowania hydrologicznego. Należało więc z jednej strony dostosować już istniejące modele do nowego środowiska, z drugiej zaś stworzyć zupełnie nowe rodzaje modeli. Wydaje się, że to ostatnie zadanie nadal jest jeszcze przed nami.

W artykule opisano możliwości wykorzystania w modelowaniu hydrologicznym programu GRASS w wersji 6.4. Jest to aktualna stabilna wersja programu. Zaprezentowano jedynie te moduły, które bezpośrednio związane są z modelowaniem hydrologicznym. Liczba modułów związanych z hydrologią pośrednio znacznie wykracza poza objętość tego artykułu. Można do nich zaliczyć na przykład całą grupę modułów wykorzystywanych do przetwarzania zobrazowań satelitarnych, które dostarczają informacje o pokryciu terenu czy wilgotności gruntu.

7.2. Matematyczne modelowanie procesów hydrologicznych

Matematyczne modele procesów hydrologicznych starają się w możliwie dokładny sposób opisać obieg wody w przyrodzie. Jest to jednak zjawisko złożone i nie do końca jeszcze poznane, a liczba parametrów, które mają wpływ na modelowane procesy, jest znaczna. Modele hydrologiczne stosowane w praktyce są więc siłą rzeczy dużym uproszczeniem świata rzeczywistego. Nie wynika to tylko z niemożności opisania równaniami matematycznymi samych procesów, ale również z problemu dostarczenia odpowiednich danych do takiego modelu. Następuje powolne przechodzenie od modeli „czarnej skrzynki” do modeli „białej skrzynki” [7]. Coraz bardziej interesuje nas zrozumienie i opisanie procesów, a nie tylko uzyskanie dobrych wyników modelowania.

Współczesne modele hydrologiczne najczęściej wykorzystują jako środowisko obliczeniowe mapy rastrowe. Środowisko to nieźle sprawdza się przy modelach procesów fizycznych o parametrach skupionych, jak i dyskretnie rozłożonych. Powstaje jednak pytanie: jakie rozdzielczości przestrzenne i czasowe należy przyjąć dla modeli poszczególnych procesów? Niestety gwałtownemu rozwojowi technik obliczeniowych nie towarzyszył równie szybki postęp w poznaniu samych procesów hydrologicznych. I nie pomogły tu wiele nawet coraz bardziej zaawansowane systemy pomiarowe, ponieważ niektórych parametrów nie wiemy jak mierzyć. Wydaje się, że rozwój narzędzi wyprzedził nieco rozwój wiedzy i mając dostępne narzędzia do obliczeń, nie do końca wiemy co liczyć należy. Jak na przykład dokonać kalibracji parametru ”początkowa wilgotność gruntu”? I co ten parametr reprezentuje w rzeczywistości?

Choć istnieją modele hydrologiczne określane mianem integralnych [7], nadal najczęściej konieczne jest poprawne opisanie poszczególnych procesów.

Aby to wykonać potrzebne są różnorodne dane:

- opad całkowity: dane ze stacji pomiarowych, dane radarowe, satelitarne, topografię terenu, . . .
- opad efektywny: dane o pokryciu terenu, rodzaju gleb, . . .
- parowanie (transpiracja): informacje o wilgotności powietrza, prędkości wiatru, rodzaju pokrycia terenu, . . .
- transformacja hydrogramu w ciekach: stany wody, przekroje poprzeczne, krzywe konsumcyjne, sieć hydrograficzna, . . .

Oczekujemy, że zadając parametry wejściowe do modeli (np. wysokość opadu), będziemy w stanie oszacować hydrogramy odpływu w dowolnych przekrojach na cieku i dokonać weryfikacji modelu. Ale nawet z tak, wydawałoby się, prostym zadaniem mamy problemy. Do dzisiaj nie potrafimy bowiem precyzyjnie zmierzyć wysokości opadu całkowitego, o opadzie efektywnym nie wspominając.

Istnieje wiele algorytmów do generowania sieci hydrograficznych na podstawie NMT (Numerycznego Modelu Terenu), który najczęściej wykorzystywany jest w postaci mapy rastrowej. Problem polega jednak na tym, że algorytmy te muszą być powiązane z algorytmami wykorzystanymi później do modelowania np. spływu powierzchniowego. Do zbudowania poprawnego hydrologicznie NMT niezbędne są dodatkowe informacje, takie jak np. przebieg wododziałów czy przebieg samych cieków.

Do obliczania kierunku spływu powierzchniowego wykorzystywane są dwa rodzaje algorytmów [6]:

- D8 – ośmiokierunkowe, wykorzystywane przez takie moduły GRASS jak r.watershed i r.terraflow,
- D-infinity – wielokierunkowe, nie definiujące sztywnej dyskretyzacji kierunkowej, wykorzystywane np. przez moduł r.flow

Sam spływ powierzchniowy może być w GRASS-ie modelowany różnymi typami algorytmów [6]:

- SFD (ang.Single Flow Direction), masa jest przekazywana w jednym kierunku [r.watershed],
- MFD (ang.Multiple Flow Direction), masa jest rozdzielana pomiędzy kilka kierunków (komórek rastra) [r.terraflow, r.topmodel]
- 2D, eksperymentalny algorytm wykorzystywany przez moduł r.sim.water.

7.3. Modele dostępne w GRASS 6.4

Praktycznie wszystkie modele hydrologiczne dostępne w GRASS pochodzą z biblioteki rastrowej. Zgodnie z przyjętą konwencją w programie, ich nazwa zaczyna się zatem od r.*.

7.3.1. r.carve

Moduł służy do modyfikacji rastrowego NMT z wykorzystaniem wektorowej mapy przebiegu cieków. Możliwe jest wygenerowanie NMT bez obszarów płaskich w obrębie cieków oraz obniżenie terenu o zadaną wartość w tych obszarach.

Autorami modułu są Bill Brown oraz Brad Douglas.

7.3.2. r.fill.dir

To jeden z podstawowych modułów wykorzystywanych na etapie pre-processingu NMT. Jego celem jest stworzenie NMT pozbawionego obszarów bezodpływowych [4]. Moduł generuje także mapę kierunków spływu w jednym z wybranych formatów (AGNPS, ANSWERS lub GRASS). Wykorzystywany jest ośmiokierunkowy algorytm spływu (D8), a mapa wynikowa jest zbliżona do mapy ekspozycji stoków.

Jedną z opcji modułu jest generowanie mapy rastrowej wskazujące obszary na warstwie NMT trudne do modyfikacji automatycznej. Każdy z takich obszarów uzyskuje unikalny identyfikator. Umożliwia to zastosowanie innych metod do wskazanych obszarów lub powtórne (sekwencyjne) uruchomienie modułu dla kolejnych map wynikowych NTM.

Tabela 7.1 Czasy przetwarzania map rastrowych algorytmami r.terraflow i r.watershed [1]

Obszar	Wymiary mapy rastrowej	r.watershed [s]	r.terraflow [s]
Kaweah	1163 x 1424	720	180
Puerto Rico	4452 x 1378	432000	480
Hawaje	6784 x 4369	1500000 (65%)	2280

Autorem algorytmu w Fortranie jest Raghavan Srinivasan z Purdue University. Wersję dla GRASS-a w języku C opracował Roger S.Miller.

7.3.3. r.sim.water

Jest to moduł do symulacji spływu powierzchniowego. Niezbędnymi danymi wejściowymi jest NMT, rastrowe mapy pierwszych pochodnych wysokości terenu [dxin, dyin], opad efektywny (jako jedna wartość [rain val] lub mapa rastrowa [rain]) oraz współczynnik szorstkości Manninga [manin]. Dodatkowo użytkownik może podać wiele innych parametrów opisanych szczegółowo w dokumentacji modułu oraz literaturze przedmiotu [5]. W wyniku generowane są mapy rastrowe głębokości [depth], przepływu [disch] oraz błędów symulacji [err].

Autorami modułu są Helena Mitasova oraz Chris Thaxton z North Carolina State University (USA) oraz Jaroslav Hofierka z GeoModel s.r.o. (Słowacja).

7.3.4. r.terraflow

Jest to wyspecjalizowany moduł zoptymalizowany do modelowania spływu dla dużych map rastrowych [1]. Optymalizacja algorytmu została przeprowadzona pod kątem operacji zapisu i odczytu (tab. 7.1). Choć w algorytmie zastosowano uznane i zaawansowane metody modelowania spływu, należy pamiętać o tym, że moduł jest zorientowany raczej na szybkość działania, niż wierne odwzorowanie spływu powierzchniowego.

Do obliczeń wymagane jest podanie jedynie rastrowego NMT. W efekcie działania modułu generowane są liczne mapy rastrowe:

- filled – NMT pozbawiony obszarów bezodpływowych,
- direction – kierunki spływu,
- swatershed – rastrowa mapa zlewni cząstkowych,
- accumulation – rastrowa mapa akumulacji spływu powierzchniowego,
- tci (ang. topographic convergence index) – współczynnik konwergencji topograficznej, obliczany jako logarytm stosunku akumulacji spływu do lokalnego spadku.

Do wyboru są dwa algorytmy modelowania spływu – SDF oraz MFD. Algorytm może w sposób wydajny przetwarzać mapy rastrowe o rozmiarze ponad 2GB.

Algorytm modelu został opracowany w ramach The TerraFlow Project realizowanego w roku 1999 w Duke University. Autorami są Lars Arge, Jeff Chase, Pat Halpin, Laura Toma, Dean Urban, Jeff Vitter oraz Rajiv Wickremesinghe. W roku 2002 algorytm zaimplementowali w GRASS-ie Lars Arge, Helena Mitasova oraz Laura Toma.

7.3.5. r.basins.fill

Moduł generuje rastrową mapę zlewni cząstkowych na podstawie rastrowego NMT. Parametrami wymaganymi są:

- number – liczba iteracji,
- c map – mapa rastrowa z unikalnymi kodami segmentów cieków; mapa tego typu może zostać wygenerowana automatycznie przez moduł r.watershed na podstawie NMT,
- t map – mapa rastrowa z przebiegiem granic zlewni cząstkowych; mapa ta niestety powinna zostać zdigitalizowana ręcznie.

Wynikowa mapa rastrowa ze zlewniami cząstkowymi będzie posiadała kodowanie zgodne z kodowaniem mapy segmentów.

Z powodu konieczności ręcznego definiowania parametru t map, funkcjonalność tego modułu jest ograniczona. Alternatywą może być moduł r.stream.basins dostępny w dodatkach, który umożliwia w pełni automatyczną realizację tego zadania.

Autorami modułu r.basins.fill są Dale White z Pennsylvania State University (USA) oraz Larry Band z University of Toronto (Kanada).

7.3.6. r.water.outlet

Moduł wyznacza granice zlewni dla podanego punktu na podstawie rastrowej mapy z kierunkami drenowania. Kierunki drenowania mogą być wygenerowane modułem r.watershed.

Autorem modułu jest Charles Ehlschlaeger z U.S. Army Construction Engineering Research Laboratory.

7.3.7. r.watershed

Jest to rozbudowany moduł do tworzenia wielu map rastrowych w oparciu o rastrowy NMT [model wysokościowy]. W przeciwieństwie do innych modułów, r.watershed do poprawnej pracy nie wymaga NMT pozbawionego obszarów bezodpływowych. Dodatkowo (opcjonalnie) mogą być podane rastrowe mapy z rzeczywistymi obszarami bezodpływowymi oraz parametrami do modelu RUSLE (ang. Revised Universal Soil Loss Equation) [8].

Generowanymi mapami rastrowymi moga być: mapa akumulacji spływu powierzchniowego [accumulation], kierunków drenowania [drainage], lokalizacja cieków [stream] i zlewni cząstkowych [basin], parametry LS [length.slope] oraz S [slope.steepness] do modelu RUSLE.

Do modelowania wykorzystywany jest algorytm D8 spływu powierzchniowego. Użytkownik ma możliwość zdefiniowania powierzchni minimalnej dla zlewni cząstkowej [threshold].

W porównaniu z modułem r.terraflow, r.watershed generuje lepsze wyniki dla obszarów z niewielkimi spadkami terenu.

Autorem modułu jest Charles Ehlschlaeger z U.S. Army Construction Engineering Research Laboratory. Aktualnymi zmianami w module, zmierzającymi do implementacji bardziej wydajnych czasowo algorytmów, zajmuje się Markus Metz.

7.3.8. r.drain

Moduł realizuje śledzenie linii spływu powierzchniowego dla zadanych punktów źródłowych. Może wykorzystywać dwa rodzaje map wejściowych – NMT lub mapę kosztów. Dla NMT wykorzystywany jest algorytm określający maksymalny spadek terenu. Mapa kosztów może zostać wygenerowana przez moduł r.cost. W przypadku mapy kosztów konieczne jest podanie mapy kierunków spływu. Obydwie metody, śledząc drogi ekstremalnych gradientów, w efekcie prowadzą do wyznaczenia lokalnych minimów, nie zaś najniższych punktów w skali całego obszaru. Z tego powodu algorytm jest czuły na zadane punkty początkowe. Punkty początkowe do poszukiwania optymalnej drogi spływu mogą być zadane bezpośrednio jako współrzędne lub w postaci punktowej mapy wektorowej. Aby uniknąć wyszukiwania jedynie lokalnych minimów, pomocne może być wcześniejsze użycie modułów r.fill.dir, r.terraflow lub r.basin.fill. Algorytm generuje błędy przy brzegach regionów obliczeniowych, ponieważ maksymalny gradient w tych obszarach przyjmowany jest zawsze w kierunku granicy regionu.

Domyślnie moduł generuje mapę logiczną spływu. Możliwe jest jednak przeniesienie wartości z NMT wzdłuż linii spływu lub też kumulacja wartości wzdłuż tej linii.

Autorem poprawionej wersji modułu jest Roger S. Miller (2001). W roku 2004 Matteo Franchi (Liceo Leonardo Da Vinci, Włochy) oraz Roberto Flor (ITC-irst, Włochy) dodali możliwość podawania punktów źródłowych w postaci mapy wektorowej.

7.3.9. r.flow

Zadaniem modułu jest generowanie wektorowych linii spływu z rastrowego NMT. Do pracy modułu wymagane jest jedynie podanie rastrowego NMT oraz opcjonalnie mapy rastrowej z obiektami będącymi barierami dla spływu powierzchniowego [barin].

Podstawowym produktem modułu jest wektorowa mapa linii spływu [flout]. Obliczane mogą być również mapy rastrowe: długości linii spływu [lgout] oraz gęstość linii spływu [dsout]. Analiza spływu może być prowadzona od źródeł do ujścia lub w przeciwnym kierunku. Moduł może generować wektorowe linie 3D zamiast standardowych 2D. Można zadać również minimalny odstęp pomiędzy liniami spływu w zakresie 1-633 komórek rastra [skip] oraz maksymalną liczbę segmentów na jedną linię spływu w zakresie 0-3144 [bound].

Moduł może być przydatny raczej przy modelowaniu erozji niż do generowania sieci hydrograficznej.

Autorami pierwotnej wersji są Maros Zlocha i Jaroslav Hofierka z Comenius University (Słowacja). Nowsze wersje przygotowali Joshua Caplan, Mark Ruesink oraz Helena Mitasova z University of Illinois (USA).

7.3.10. r.lake

Moduł umożliwia symulowanie zbiornika retencyjnego lub jeziora. Na podstawie NMT oraz zadanego poziomu zwierciadła wody [wl] generowany jest zasięg zbiornika. Konieczne jest wskazanie punktu początkowego wypełniania terenu [xy] lub mapy rastrowej z takimi punktami [seed]. Wynik jest zapisywany jako mapa rastrowa [lake] z obliczonymi głębokościami zalewu. Obliczana jest jednocześnie powierzchnia zwierciadła wody oraz pojemność zbiornika. Cały obszar mapy jest wczytywany do pamięcie operacyjnej, więc moduł nie jest przystosowany do pracy z dużymi mapami.

Autorem modułu jest Maris Nartiss.

7.3.11. r.topmodel

Jest to implementacja modelu TOPMODEL [2] , który jest dobrze udokumentowanym matematycznym modelem procesów fizycznych o parametrach dyskretnie rozłożonych lub częściowo dyskretnie rozłożonych. Jednym z podstawowych parametrów modelu jest wskaźnik topograficzny wyznaczany przez moduł r.topidx. Zakłada się, że obszary o zbliżonej wartości tego wskaźnika posiadają zbliżone charakterystyki hydrologiczne.

Obszary o zbliżonych wartościach są grupowane i traktowane jako jednolite jednostki.

$$topidx = \ln(a) \cdot \tan(\beta)$$

gdzie

a - powierzchnia obszaru zasilającego,

β - lokalny spadek terenu.

Serie czasowe opadu oraz ewapotranspiracji potencjalnej, będące wymaganymi danymi wejściowymi, przekazywane są do modelu w pliku tekstowym. Dodatkowo wymagany jest plik tekstowy z parametrami pracy modelu, stworzony w ściśle określonym formacie. W pliku tym zawarte są dane dotyczące charakterystyki zbiornika strefy korzeniowej oraz strefy aeracji, prędkości transformacji fal w ciekach oraz wiele innych.

Jako podstawę moduł wykorzystuje NMT pozbawiony obszarów bezodpływowych. Mapa taka może zostać stworzona w ramach modelu poprzez wywołanie modułu r.fill.dir lub podana jako mapa wejściowa. Podobna sytuacja odnosi się do parametru topidx. Może on być liczony lub podawany jako mapa wejściowa.

Model wymaga kalibracji parametrów. Podanie w pliku wejściowym danych o przepływach obserwowanych umożliwia bezpośrednią weryfikację działania modelu.

Wersja dla GRASS-a została przygotowana w roku 2000 jako praca magisterska przez Huidae Cho z Kyungpook National University (Korea Południowa).

7.4. Modele dostępne jako dodatki do GRASS 6.4

Istnieje kilka modułów, które mogą być bezpośrednio wykorzystane w modelowaniu hydrologicznym, lecz nie są dostępne w standardowej instalacji GRASS-a. Ich kody źródłowe są dostępne na stronach projektu GRASS, ale do pracy z GRASS-em wymagają oddzielnej instalacji. Są to tak zwane moduły dodatkowe (ang. add-on).

7.4.1. HydroFOSS

Moduł HydroFOSS to model procesów fizycznych o parametrach dyskretnie rozłożonych. Uwzględnia wiele procesów, w tym m.in. promieniowanie słoneczne, ewapotranspirację, topnienie śniegu oraz intercepcję [3].

Moduł został stworzony przez Massimiliano Cannata z Politechniki w Mediolanie (Włochy) w roku 2006 jako praca doktorska. Dzięki temu w sposób wyjątkowo szczegółowy opisane zostały podstawy merytoryczne modelu oraz wyniki jego pracy.

7.4.2. r.stream.*

Jest to grupa modułów do wyznaczania parametrów Hortona w zlewniach. Pierwszy z nich [r.stream.order] służy do wyznaczeniu rzędów cieków według czterech najpopularniejszych klasyfikacji (Strahlera, Hortona, Shreve'a, Hacka).

Drugi [r.stream.basins] wyznacza zlewnie cząstkowe zgodnie z zadanymi regułami.

Trzeci [r.stream.distance] służy do tworzenia map odległości od cieków oraz ujść cieków.

Czwarty [r.stream.stats] umożliwia obliczenie wybranych statystyk związanych ze zlewniami. Dostępne są jeszcze moduły r.stream.angle, r.stream.del, r.stream.extract, r.stream.pos.

Autorami modułów jest Jarosław Jasiewicz z Uniwersytetu im. Adama Mickiewicza w Poznaniu oraz Markus Metz.

7.4.3. GIPE

GIPE (Grass Image Processing Environment) jest konglomeratem wielu modułów.

Oprócz modelu r.hydro.CASC2D, pakiet zawiera wiele modułów do przetwarzania zobrazowań satelitarnych. Są to przykładowo moduły do wyznaczania ewapotranspiracji potencjalnej [i.evapo.potrad, i.evapo.SENAY].

Autorem większości modułów pakietu GIPE jest Yann Chemin.

7.5. Wnioski

Najczęściej modelowanie hydrologiczne z wykorzystaniam GIS rozumiane jest jako pre-processing. Większość modułów dostępnych w menu GRASS pod kategorią hydrologia, to moduły do wyznaczania charakterystyk topograficznych w oparciu o numeryczny model terenu.

Dla potrzeb hydrologii mogą też być wykorzystywane moduły umożliwiające np. wyznaczenie statystyk zlewni cząstkowych dla modeli o parametrach skupionych czy przygotowanie map (pokrycie terenu, gleby, uwilgotnienie) w oparciu o zobrazowania satelitarne.

Nieliczne z dostępnych modeli prezentują bardziej kompleksowe podejście do problematyki modelowania hydrologicznego. Należy do nich zaliczyć r.sim.water, r.topmodel, HydroFOSS oraz GIPE.

Modele hydrologiczne udostępniane w ramach projektu GRASS tworzone są w sposób nieskoordynowany i mają dość krótkie okresy aktywności ich twórców. Raczej tworzone są nowe grupy modułów, niż doskonalone już istniejące.

W tej chwili praktycznie brak jest kompleksowego modelu hydrologicznego aktywnie rozwijanego w ramach projektu GRASS. Większość prac, choć prowadzona na wysokim poziomie merytorycznym, dotyczy jedynie wybranych fragmentów problematyki modelowania hydrologicznego.

Literatura

[1] Arge L., Chase J.S., Halpin P.N., Toma L., Vitter J.S., Urban D., Wickremesinghe R., 2003, *Flow computation on massive grid terrains*, [w:] GeoInformatica, International Journal on Advances of Computer Science for Geographic Information Systems, 7(4), s. 283-313

[2] Beven K., Lamb R., Quinn P., Romanowicz R., Freer J., 1995, *TOPMODEL*, [w:] Singh V.P. (Ed.). Computer Models of Watershed Hydrology. Water Resources Publications, s. 627-668

[3] Cannata M., 2006, *A GIS embedded approach for Free & Open Source Hydrological Modelling*. PhD thesis, Department of Geodesy and Geomatics, Polytechnic of Milan, Italy.

[4] Jenson S.K., Domingue J.O., 1988, *Software Tools to Extract Structure from Digital Elevation Data for Geographic Information System Analysis,* Photogrammetric engineering and remote sensing, Vol. 54, No. 11, November 1988, s. 1593-1600

[5] Mitasova H., Thaxton C., Hofierka J., McLaughlin R., Moore A., Mitas L., 2004, *Path sampling method for modeling overland water flow, sediment transport and short term terrain evolution in Open Source GIS*, [w:] Miller C.T., Farthing M.W., Gray V.G., Pinder G.F. (Eds.), Proceedings of the XVth International Conference on Computational Methods in Water Resources, Chapel Hill, NC, USA, Elsevier, s. 1479–1490.

[6] Neteler M., Mitasova H., 2008, *Open source GIS: A GRASS GIS approach. Third Edition*, The International Series in Engineering and Computer Sciences: Volume 773. Springer New York Inc. ss. 406

[7] Soczyńska U. (red.), 1997, *Hydrologia dynamiczna*, Wydawnictwo naukowe PWN, Warszawa.\

[8] Weltz M.A., Renard K.G., Simanton J.R., 1987, *Revised Univversal Soil Loss Equation for Western Rangelands*, U.S.A./Mexico Symposium of Strategies for Classification and Management of Native Vegetation for Food Production in Arid Zones, Tuscon, 12-16 październik 1987

Źródła internetowe

[9] Strona domowa projektu GRASS http://grass.osgeo.org

8. Prezentacja kartograficzna w systemie GRASS

Adam Górecki, Jacek Ślopek

8.1. Wstęp

Systemy GIS, od chwili pojawienia się pierwszych tego typu programów na rynku około 40-50 lat temu, znalazły wiele zastosowań w licznych dziedzinach życia i nauki, wykraczających poza pierwotne tworzenie baz danych o zasobach naturalnych. Niemniej od samego początku historia oprogramowania typu GIS nierozerwalnie związana jest z dziejami kartografii. Zanim pojawiły się pierwsze systemy GIS podejmowano wiele wysiłków, by zautomatyzować proces kartograficzny. Ręczne tworzenie map starano się przyspieszyć przy wykorzystaniu maszyn cyfrowych. Jako jeden z przykładów takich działań można podać opracowanie ponad 2000 map przez brytyjskich botaników [4] opracowujących „Atlas of the British Flora". Na kanwie takich doświadczeń w latach sześćdziesiątych XX w. zaczęto tworzyć podwaliny naukowe pod przyszłe zinformatyzowane systemy GIS. Poprzez rozwój techniki komputerowej w latach 80-tych i rozpowszechnienie się komputerów osobistych, nastąpiło także upowszechnienie się systemów GIS [2].

Z powodu wspomnianych związków historycznych, oprogramowanie GIS jest najczęściej postrzegane jako oprogramowanie służące do obsługi map cyfrowych. Niemniej, mimo iż systemy GIS w pewnych zastosowaniach mogą być utożsamiane z komputerową kartografią, to istnieją jednak pomiędzy tymi rodzajami oprogramowania znaczące różnice, o czym należy pamiętać. Na podstawie produktu kartograficznego możemy uzyskać bowiem jedynie informacje wynikające z reprezentacji graficznej poszczególnych elementów mapy. System GIS pozwala natomiast na uzyskanie dodatkowych informacji o każdym z obiektów zawartych na mapie oraz pozwala na zadawanie złożonych zapytań dotyczących właściwości opisujących te obiekty oraz relacji pomiędzy nimi [2].

W niniejszej pracy omówiona zostanie funkcjonalność kartograficzna systemu GRASS, który ujrzał światło dzienne w połowie lat 80-tych XX wieku.

Systemy GIS nie mogą się oczywiście obyć bez choćby minimalnej funkcjonalności kartograficznej, a więc możliwości prezentacji (wizualizacji) danych zawartych w bazie danych w postaci map, czy to na ekranie monitora, czy też po wydrukowaniu ich na papierze.

Prezentacja kartograficzna realizowana jest na mapie w postaci umownych znaków graficznych, nazywanych znakami kartograficznymi, które rozmieszczone są w treści mapy jako znaki punktowe (rozmieszczenie dyskretne), znaki liniowe (rozmieszczenie liniowe) oraz znaki powierzchniowe (występowanie ciągłe i wyspowe). Znaki kartograficzne oraz napisy na mapie są zlokalizowane według współrzędnych kartezjańskich lub geograficznych i przedstawione według zmiennych graficznych: wielkości, jasności, ziarnistości, koloru, orientacji i kształtu w odpowiedniej hierarchii wizualnej [1, 3].

Popularność prezentacji informacji przestrzennej na mapie wynika nie tylko z powszechności technologii wizualnych, ale przed wszystkim z powodu szybkości i skuteczności podejmowania decyzji na podstawie graficznej prezentacji danych.

Przykładowo, aplikacja ArcMap będąca elementem bazowego pakietu ArcGIS firmy ESRI pozwala przy redagowaniu kompozycji i drukowaniu map na tworzenie [11]: szablonów map, ramek oddzielających treść mapy od marginesów, siatki geograficznej,

siatki kilometrowej, siatki indeksowej, wielu ramek danych, elementów mapy powiązanych z ramką danych: strzałki północy, skala liczbowa, podziałka liniowa, skalowalna legenda, napisy oraz tytuły, obwódki, rysunki.

Narzędzie do poprawy jakości i szybkości drukowania oraz eksportu map, umożliwiające zaawansowany wydruk map nosi nazwę ArcPublisher. Charakterystyczna dla wielu programów komercyjnych, zamknięta i zintegrowana struktura dostarcza narzędzi potrzebnych do wszystkich zadań typu GIS, takich jak: wprowadzenie i edycja danych, produkcja kartograficzna, zarządzanie danymi, wykonywanie analiz przestrzennych, obsługa metadanych oraz udostępnianie danych i aplikacji poprzez Internet. Produkt firmy ESRI nie jest jednakże pozbawiony wad. Zamknięty charakter ArcGIS uniemożliwia ingerencję w oprogramowanie i jest przeszkodą w przypadku wielu problemów wymagających rozwiązań eksperckich.

Aby wydrukowana mapa przypominała produkt kartograficzny konieczne jest wyposażenie systemu GIS w funkcje o profilu kartograficznym. System GRASS dysponuje odpowiednimi funkcjami, pozwalającymi na wizualizację danych zgromadzonych w jego bazie, czy to w postaci plików graficznych, do wykorzystania w prezentacjach, lub też na stronach www (np. display driver PNG), a także pozwalających na wygenerowanie plików w formacie PostScript, będącym standardem dla maszyn drukarskich. Polecenia odpowiedzialne za generowanie plików w formacie PostScript pozwalają uzyskać mapę wynikową spełniającą warunki produktów kartograficznych.

8.2. Wizualizacja danych w systemie GRASS

8.2.1. Sterownik graficzny PNG

System GRASS ma możliwość przekierowania graficznych wyników pracy na kilka rodzajów wyjść. Najczęściej wykorzystywany jest monitor graficzny, ale niekiedy zachodzi potrzeba utrwalenia wyników naszej pracy w systemie, w postaci plików graficznych, do których można będzie powrócić w przyszłości lub też wykorzystać je jako ilustracje w prezentacjach, artykułach, czy też na stronach internetowych. Aby móc skorzystać z tej funkcjonalności użytkownik systemu nie musi poznawać złożonej składni kolejnych poleceń, czy też uczyć się na nowo sposobów na generowanie pożądanej mapy wynikowej. Użycie w systemie GRASS tzw. display drivers jest niemal identyczne jak wyświetlanie danych na monitorze, w trakcie codziennej pracy z systemem. Jedyne co odróżnia zastosowanie wyjścia do pliku graficznego, od zastosowania monitora graficznego, to fakt wykonania kilku dodatkowych czynności. Przekierowanie wyników obliczeń w systemie GRASS do pliku następuje w czterech krokach:

1. Ustawienie zmiennych systemowych określających parametry wynikowego pliku graficznego
2. Uruchomienie przekierowania do pliku PNG,
3. Użycie poleceń systemu GRASS: ustalanie regionu, rysowanie map, elementów graficznych takich jak: skala, legenda, itp.,
4. Zakończenie przekierowania do plików PNG i zapisanie wyników na dysku twardym komputera.

Wszystkie te czynności można wykonywać krok po kroku wydając kolejne komendy systemowi, ale można również zautomatyzować swoją pracę. System GRASS dopuszcza bowiem uruchamianie mini-programów – skryptów powłoki systemowej – które wykonują te czynności za nas.

8.2.2. Ustawienie zmiennych systemowych

Na tym etapie użytkownik musi zdecydować, jaki format nada mapie wynikowej. Zanim rozpocznie się właściwe wysyłanie danych do pliku graficznego trzeba określić m. in. rozmiar, nazwę, tło, paletę barwną, przezroczystość tła, czy kompresję

pliku. Uzyskuje się to za pomocą ustawiania zmiennych systemowych (w systemie Linux za pomocą polecania export). System GRASS korzysta z następujących zmiennych systemowych:

- GRASS_WIDTH (szerokość w pikselach pliku wynikowego),
- GRASS_HEIGHT (wysokość w pikselach pliku wynikowego),
- GRASS_PNGFILE (nazwa pliku wynikowego),
- GRASS_BACKGROUNDCOLOR (kolor tła dla pliku wynikowego),
- GRASS_TRANSPARENT (włączenie przezroczystości tła – kanału alfa),
- GRASS_TRUECOLOR (włączenie palety barwnej Truecolor – obsługi wyświetlania 16777216 kolorów),
- GRASS_PNG_COMPRESSION (włączenie kompresji pliku o zadanym stopniu),
- GRASS_PNG_AUTO_WRITE (włączenie automatycznego zapisu pliku po każdym poleceniu systemu wydanym przez użytkownika, a nie dopiero na samym końcu pracy ze sterownikiem graficznym,
- GRASS_PNG_READ (włączenie odczytu z już istniejącego pliku i dopisywania informacji do tego pliku)

8.2.3. Uruchomienie i zatrzymanie przekierowania wyników pracy do pliku PNG

Uruchomienie i zatrzymanie przekierowania do pliku graficznego następuje w niezwykle prosty sposób. Można to wykonać za pomocą polecenia d.mon:

```
d.mon start=PNG
d.mon stop=PNG
```

gdzie d.mon jest poleceniem sterującym monitorem graficznym, a PNG jest nazwą używanego sterownika. Odpowiednie opcje (start/stop) powodują otwarcie, bądź zamknięcie monitora.

8.2.4. Polecenia systemu GRASS

Polecenia używane w trakcie generowania mapy wynikowej są takie same jak polecenia systemu GRASS, pozwalające umieszczać na ekranie dane rastrowe (d.rast), wektorowe (d.vect) i elementy charakterystyczne dla map, tj. legendy (d.legend), skalę liniową i strzałki północy (d.barscale), czy też opisy tekstowe (d.text). Pozwalają też na zmianę czcionek wykorzystywanych w opisach tekstowych (d.font). Dostępne dla użytkownika jest także polecenie sterujące regionem obliczeniowym (g.region), pozwalające określić zasięg mapy wynikowej w bieżącej lokalizacji.

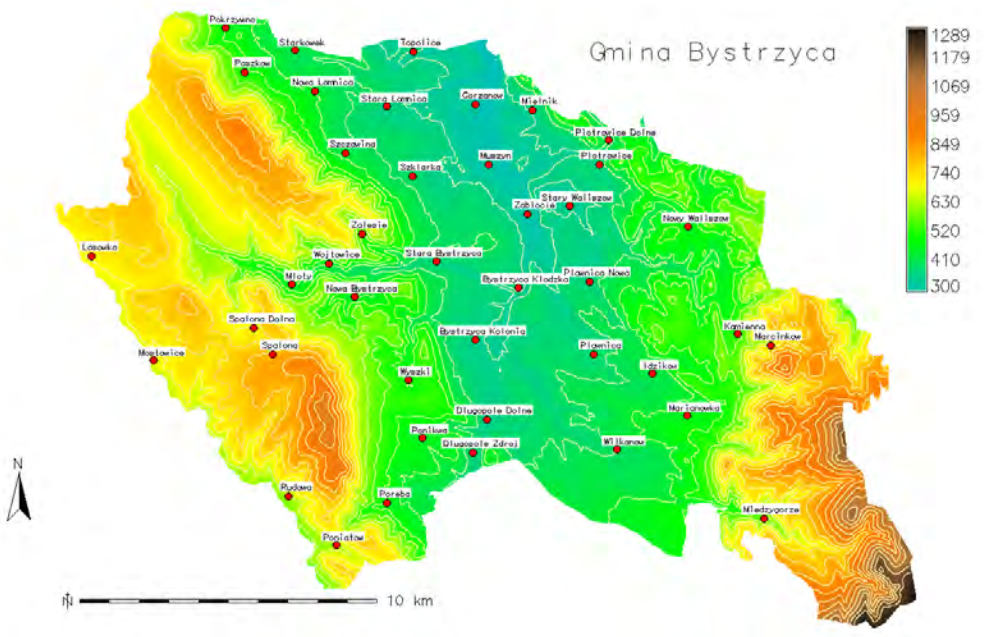
8.2.5. Usprawnienie pracy (automatyzacja za pomocą skryptów powłoki)

Przy skomplikowanych mapach lub w sytuacji wymagającej stworzenia wielu podobnych map różniących się w niewielkim stopniu, warto skorzystać z jeszcze jednej cechy systemu GRASS, a mianowicie z obsługi skryptów systemowych. Są to miniprogramy pisane w języku powłoki systemowej np. bash. Dzięki nim uzyskać można automatyzację swojej pracy.

Skrypt taki nie jest niczym innym, jak plikiem tekstowym z wpisaną wewnątrz, w kolejnych liniach, listą poleceń, które mówią systemowi GRASS co i w jakiej kolejności ma wykonać.

Przykładowy skrypt generujący mapę w systemie GRASS (rys. 8.1)

```
#!/bin/sh

##################################
# Ustawiane są zmienne systemowe
##################################

export GRASS_WIDTH=1280
export GRASS_HEIGHT=800
export GRASS_BACKGROUNDCOLOR=FFFFFF
export GRASS_PNGFILE=
   dtm_bystrzyca_m.png
export GRASS_TRUECOLOR=TRUE
export GRASS_PNGCOMPRESSION=0

##################################
# Rozpoczyna się przekierowanie
# danych do pliku
##################################

d.mon start=PNG

##################################
# Wykonywane są kolejne polecenia
# systemu GRASS.
##################################

g.region gmina_bystrzyca
d.erase
d.rast dtm_bystrzyca_m
```



Rys. 8.1 Mapa powstała po wykonaniu skryptu.

```
d.vect color=255:255:180
   map=poziomice_50_bystrzyca
d.barscale --q at=10.0,90.0
d.barscale --q -n at=5.0,70.0
d.legend --q map=dtm_bystrzyca_m
   at=55.0,95.0,90.0,92.0 color=black
   labelnum=10 range=300.0,1289.0
d.text text="Gmina Bystrzyca" size=4
   color=black at=60.0,90.0
d.vect map=miejscowosci size=10
   fcolor=red color=black
   icon=basic/circle
v.label map=miejscowosci
   labels=nazwy_miejscowosci
   column=str_1 xoffset=10 yoffset=10
   reference=center size=150
   fontsize=8 color=black
   background=white opaque=yes
d.labels labels=nazwy_miejscowosci

#################################
# Zatrzymywany jest strumień
# danych do pliku graficznego.
# Plik zostaje zapisany.
#################################

d.mon stop=PNG
```

8.3. Generowanie mapy w formacie PostScript za pomocą polecenia ps.map

System GRASS jest wyposażony w możliwości prezentacji graficznej wychodzące poza prosty system wyświetlania map cyfrowych. Twórcy oprogramowania zadbali o to, by można było wydrukować mapę papierową, jak najbardziej zbliżoną do profesjonalnych produktów kartograficznych. Odpowiedzialny jest za to moduł obsługi języka PostScript o nazwie ps.map, który generuje pliki w formacie PS (ang. *PostScript*) lub EPS (ang. *Encapsulated PostScript*) [7, 9]. Ponieważ zdecydowana większość użytkowników nie ma w codziennej pracy choćby minimalnej styczności z programowaniem w języku PostScript i zapewne stworzenie pliku w tym języku stanowiło by sporą przeszkodę, ps.map stanowi platformę pośredniczącą pomiędzy człowiekiem a maszyną drukarską. Użytkownik musi stworzyć plik złożony ze specjalnych poleceń, które ps.map zinterpretuje i przetłumaczy na język PostScript zrozumiały dla urządzeń drukarskich.

Przykładowy skrypt dla polecenia ps.map. Wynikowa mapa znajduje się na rysunku 8.2.

```
paper      a4
   left 0.4
   right 0.4
   bottom 0.4
   top 0.4
end

border     y
   width 2.5
   color black
end

scale 1:125000

raster dtm_bystrzyca_m

vlines poz_50_bystrzyca
   color 230:180:115
```

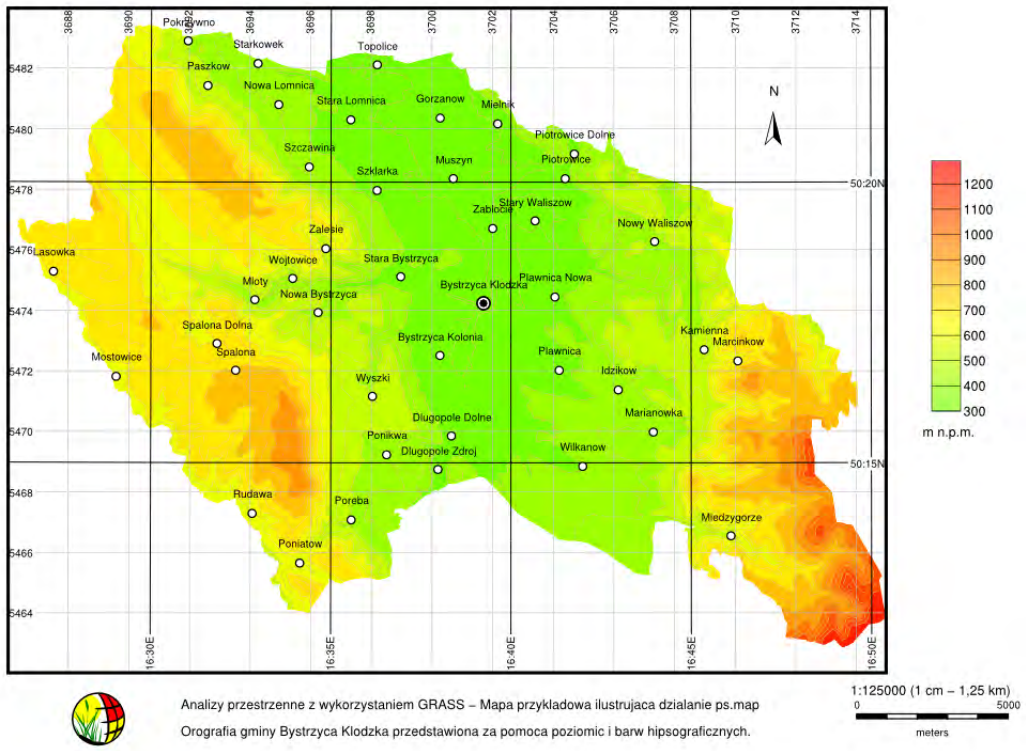
```
   width 0.4
end

vpoints miejscowosci
   type point
   where str_1<>'Bystrzyca Klodzka'
   color black
   fcolor white
   symbol basic/circle
   size 6
end

vpoints miejscowosci
   type point
   where str_1='Bystrzyca Klodzka'
   color black
   fcolor white
   symbol kartografia/miasto_1
   size 10
end

labels nazwy_miejscowosci_ps
end

colortable y
   where 10.0 2.0
   raster dtm_bystrzyca_m
   range 300 1290
   width 0.3
   height 2.6
   color black
   tickbar y
end

text 3717000 5470000 m n.p.m.
  background white
end

grid      2000
   color grey
   numbers 1 black
   width 0.5
end

geogrid 5 m
   color black
   numbers 1 black
   width 0.8
end

scalebar f
    where 10.0 7.8
    length 5000
    height 0.05
    segment 5
    numbers 5
    units meters
end

text 3716400 5461500 1:125000
   (1 cm - 1,25 km)
  background white
end

point 3711250 5480000
   symbol extra/n_arrow1
   fcolor black
   size 12
end

text 3711250 5481250 N
end

text 3701250 5460500 Analizy
   przestrzenne z wykorzystaniem
   GRASS - Mapa przykladowa
   ilustrujaca dzialanie
   ps.map\n\nOrografia gminy
   Bystrzyca Klodzka przedstawiona za
   pomoca poziomic i barw
   hipsograficznych.
  background white
end

eps 3689000 5460500
  epsfile WGUG_warsztaty_logo.eps
  scale 0.2
end
```

W aktualnej, stabilnej wersji systemu GRASS (oznaczonej numerem 6.4) użytkownik ma dostęp do zestawu ponad trzydziestu poleceń dla ps.map. Polecenia te są odpowiedzialne za wyświetlanie i nadawanie pożądanego wyglądu poszczególnym elementom mapy. Kontrolują one np. pojawienie się na mapie warstwy rastrowej, warstw wektorowych, siatki współrzędnych geograficznych, siatki odległości, legendy, opisów tekstowych, skali liniowej, symboli graficznych, a nawet zewnętrznych plików graficznych.

Można w ten sposób umieścić na mapie np. wykresy w formacie EPS wygenerowane w innej aplikacji, choćby logotyp uczelni. Polecenia te określają położenie wszystkich elementów. Określają one również skalę mapy.

Jako pierwsze w skrypcie pojawia się polecenie paper, w którym użytkownik może określić, jakiego rozmiaru papier będzie podkładem pod docelową mapę, a także wybrać, jaki obszar będzie zadrukowywany (poprzez określenie marginesów strony). Polecenie, które ma więcej niż jedną opcję wstawiane jest w skrypcie jako blok: polecenie – opcje – klauzula „end”. Nawet jeśli nie wszystkie opcje wypisywane są w skrypcie, to takie polecenie należy zamknąć klauzulą „end”. Wyjątek stanowią polecenie raster (pozwalające na dodanie do mapy warstwy rastrowej systemu GRASS) oraz polecenie scale (określające docelową skalę mapy).

Polecenie border określa, czy wokół mapy narysowana zostanie ramka i jakiej grubości i koloru ona będzie (w powyższym przykładzie jest to ramka czarna, o grubości 2.5 punktu drukarskiego, który jest definiowany jako 1/72 cala, a więc 0.352(7) mm).



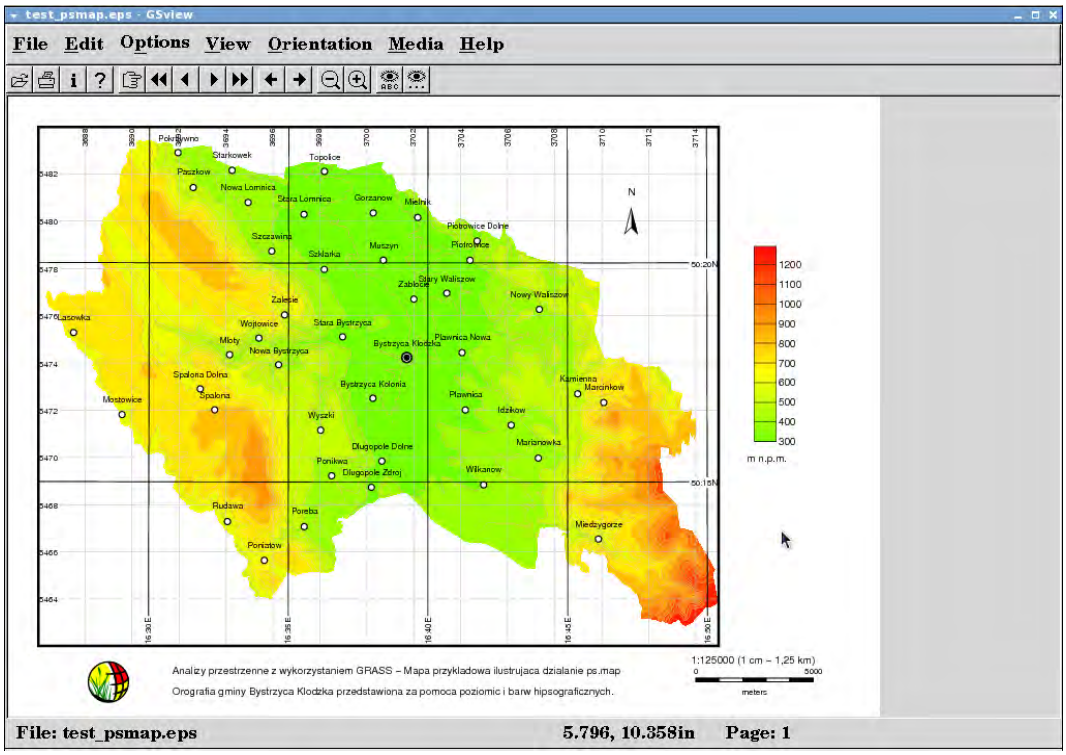
Rys. 8.2 Wynikowa mapa uzyskana za pomocą polecenia ps.map i przykładowego skryptu.

Polecenia odpowiedzialne za umieszczenie wektorowych warstw systemu GRASS to vlines i vpoints (pozwalające odpowiednio umieścić na mapie warstwę liniową i punktową). W poleceniach tych użytkownik ma możliwość określenia grubości, kolorów, jakimi będą rysowane linie, a także wskazania, z jakich symboli chce skorzystać podczas oznaczania punktów.

Dodatkowo, polecenie vpoints pozwala na skorzystanie z instrukcji będącej odpowiednikiem zapytania do bazy danych (where), dzięki któremu można określić, które elementy warstwy punktowej mają być wyrysowane, co pozwala na wybór indywidualnych symboli dla każdego punktu na warstwie osobno. Jeśli w systemie GRASS został utworzony plik z etykietami dla warstwy wektorowej (poleceniem v.label) polecenie labels w ps.map pozwala je wyświetlić, zgodnie ze zdefiniowanym wyglądem.

Polecenie colortable odpowiedzialne jest natomiast za umieszczenie na mapie skali barwnej powiązanej z wczytaną wcześniej warstwą rastrową. Występujące w skrypcie polecenia text i point pozwalają odpowiednio wstawić na mapę w określonych położeniach łańcuchy tekstowe lub symbole. W ten sposób przykładowa mapa została opisana, a we wskazanym miejscu umieszczono symbol strzałki północy. W podobny sposób, za pomocą polecenia eps, na mapie umieszczone zostało przykładowe logo. Polecenia grid i geogrid pozwalają na naniesienie na mapie siatek – kilometrowej i geograficznej, natomiast dzięki poleceniu scalebar umieszcza się podziałkę liniową. Całość skryptu musi zostać zakończona dodatkową klauzulą „end”, co pozwoli interpreterowi na rozpoczęcie pracy i zapisanie pliku w formacie PostScript.

Zestaw poleceń i parametrów programu ps.map może być jednak uznany za niekompletny z punktu widzenia użytkownika nastawionego na zastosowanie systemu GRASS jako narzędzia do tworzenia map. Wynika to ze sposobu postrzegania tego systemu przez jego twórców.



Rys. 7.3 Podgląd mapy w programie GSView

U podstaw jego stworzenia leży bowiem założenie, iż GRASS ma być narzędziem do analiz przestrzennych i modelowania, a nie programem do komputerowej kartografii. Wszelkie braki systemu GRASS w tym zakresie można uzupełnić jednak poprzez edycję wynikowych plików programu ps.map w zewnętrznych aplikacjach, pozwalających na wizualizację i obróbkę plików w formacie PostScript.

8.4. Aplikacje wspomagające generowanie map

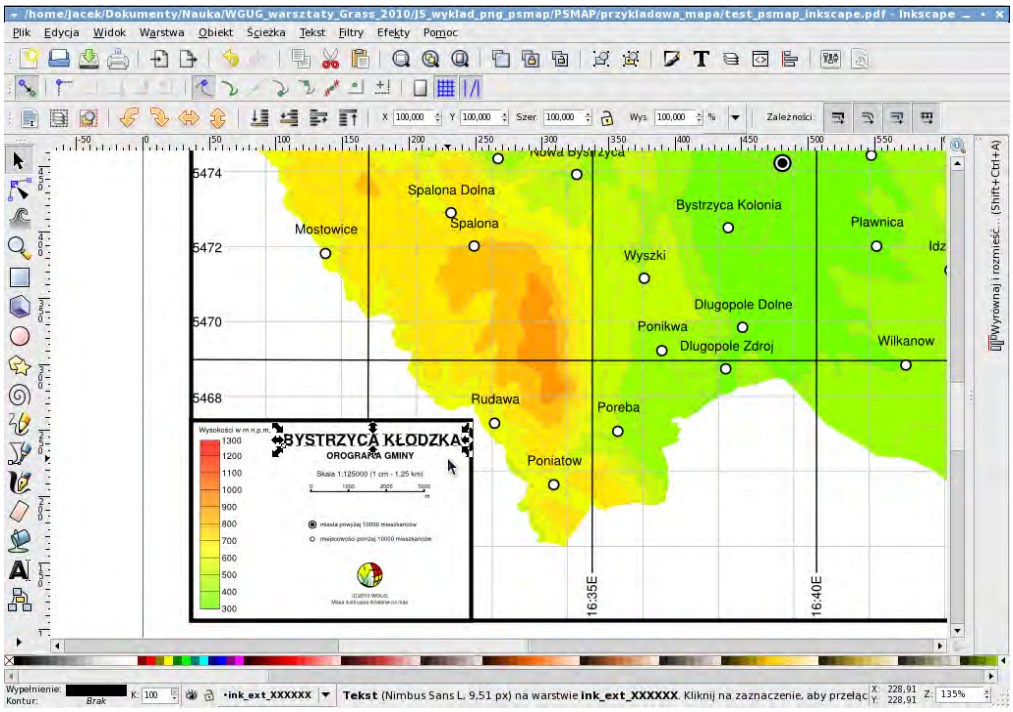
8.4.1. GSView

Bardzo wygodnym narzędziem służącym do podglądu map tworzonych przy użyciu ps.map jest program GSView [6], który służy do podglądu i konwersji plików zapisanych w formacie PostScript. GSView współpracuje ściśle z bibliotekami interpretera języka PostScript – ghostscript. Poza podglądem plików pozwala również na konwersję plików np. do formatu PDF. Mając uruchomiony jednocześnie program GSView i wywołując polecenie ps.map z linii komend systemu GRASS, użytkownik ma do dyspozycji niemal natychmiastowy podgląd edytowanej mapy i widzi efekty zmian wprowadzonych do pliku z poleceniami. Aplikacja ta podaje również użytkownikowi informacje na temat położenia kursora nad mapą, co jest dość istotne w rozmieszczaniu elementów na mapie. W systemie GRASS położenie większości elementów na mapie ustalane jest względem jednego z narożników mapy.

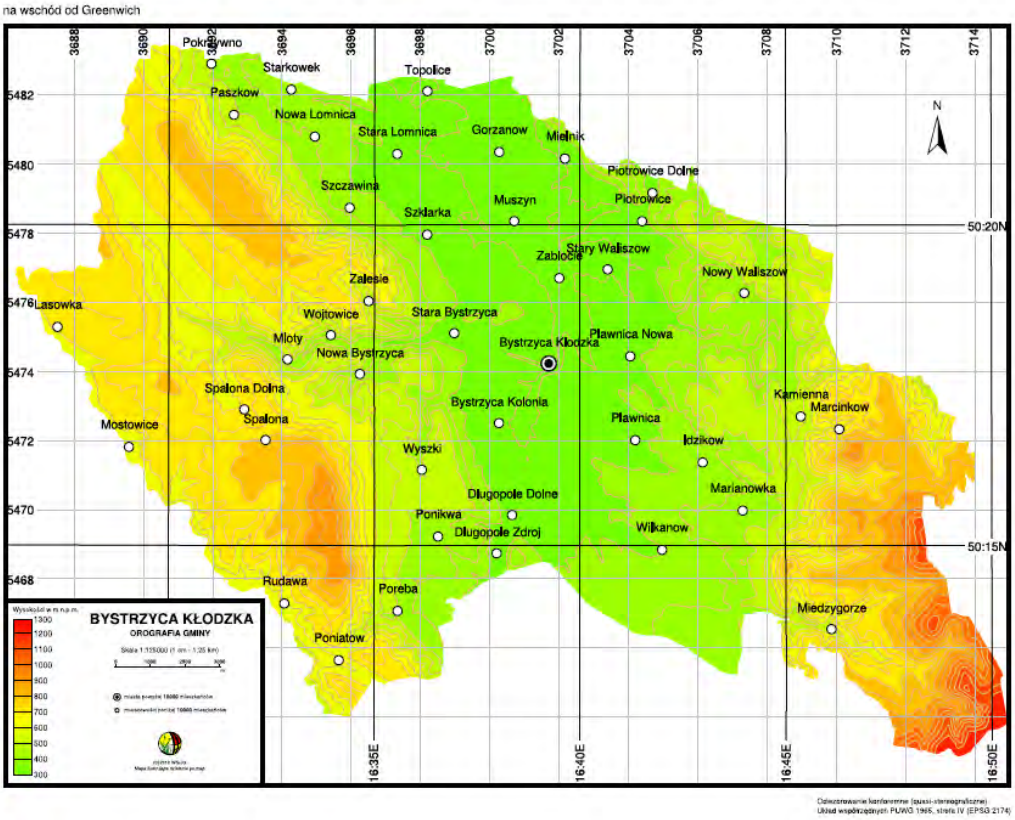
Jednostkami odległości są najczęściej cale (czasem odległości wyrażane są w procentach długości boku mapy), co w przypadku użytkowników pracujących z jednostkami układu SI, stanowiło pewną przeszkodę w sprawnym i szybkim ustalaniu właściwych wartości parametrów.

8.4.2. Inkscape

Program Inkscape jest programem do obróbki grafiki wektorowej [5]. Inkscape bazuje na stworzonym przez W3C (ang. *World Wide Web Consortium*) formacie plików SVG (ang. *Scalable Vector Graphics*) opisującym statyczną i animowaną grafikę dwuwymiarową.



Rys. 8.4 Edycja mapy w programie Inkscape



Rys. 8.5 Przykładowa mapa uzyskana za pomocą polecenia ps.map, po zmianach w programie Inkscape.

Program pozwala użytkownikowi na pracę z krzywymi, obiektami, tekstem. Pozwala również na import grafiki rastrowej, tworzenie gradientów, grupowania obiektów, czy obsługę tzw. barwnego kanału alfa dla uzyskania przeźroczystości wypełnienia obiektów. Lista formatów wektorowych obsługiwanych przez program nie kończy się na SVG. Dzięki możliwości obsługi formatu PS i EPS nadaje się on świetnie do edycji map wygenerowanych w systemie GRASS. Użytkownik wczytawszy plik PostScript do programu Inkscape uzyskuje całkowitą kontrolę nad wszystkimi elementami znajdującymi się na mapie zapisanej w tym pliku. Uzyskuje się możliwość przemieszczenia, zmiany wielkości, koloru obiektów. Możliwe jest również usuwanie elementów, czy też dodawanie takich części map, których ps.map w systemie GRASS nie generuje (np. dodatkowe ramki wokół mapy). Ponieważ program ten pracuje w polskojęzycznym środowisku, zyskujemy dodatkowo możliwość wstawiania opisów tekstowych pisanych w języku polskim (ps.map domyślnie obsługuje zachodnie kodowanie Latin-1). Zarówno GSView, jak i Inkscape są dostępne na licencji GPL, co oznacza, że korzystać możemy z nich całkowicie za darmo, podobnie jak z samego systemu GRASS.

8.5. Generowanie plików PostScript za pomocą polecenia ps.output

GRASS jest systemem pozwalającym na rozszerzanie podstawowej wersji (instalacji) o dodatkowe wyspecjalizowane moduły, które powstawały niezależnie od trzonu systemu. Użytkownik może zainstalować specjalne dodatki (tzw. Add-On), które zwiększają ilość funkcji systemu i tym samym rozszerzają jego funkcjonalność w trakcie przygotowania map. Dodatek, który trzeba doinstalować do systemu GRASS, odpowiedzialny za zwiększenie liczby funkcji kartograficznych, nosi nazwę ps.output. Jego autorem jest E. Jorge Tizado [8, 10].

Dodatek ps.output został stworzony, by rozszerzyć możliwości prezentacji kartograficznej w stosunku do programu ps.map o m. in. zgodne ze standardami obramowania map, czy np. skale liniowe. Pozwala on również na dużo większą kontrolę nad elementami graficznymi i zwiększa precyzję układania poszczególnych elementów mapy (pozycjonowanie do dziesiątych części milimetra).

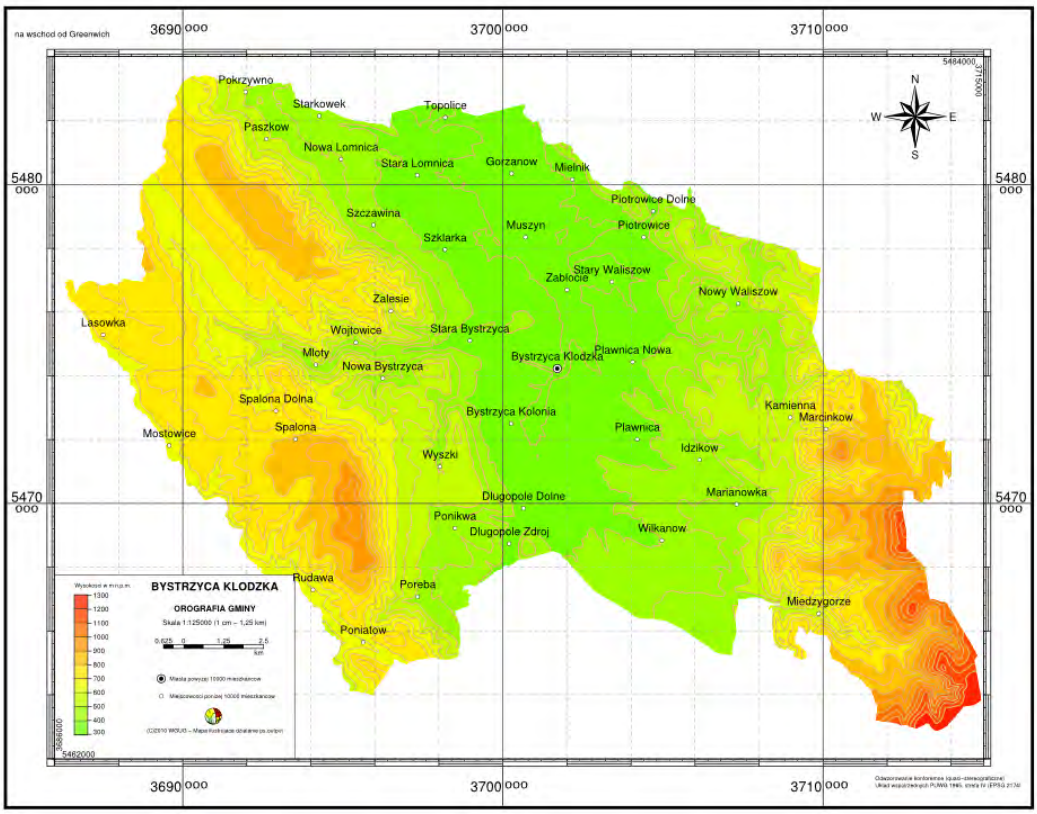
Wykorzystanie ps.output jest zbliżone do wykorzystanie polecenia ps.map. Użytkownik, by stworzyć plik w formacie PostScript zawierający polecenia dla drukarki, czy maszyny drukarskiej, musi stworzyć plik tekstowy z poleceniami, które zostaną zinterpretowane w następnej kolejności przez polecenie ps.output i przetłumaczone do języka PostScript. Różnice pomiędzy ps.map i ps.output polegają zasadniczo na nieco innej składni poleceń.

Rozszerzenie to pozwala dodatkowo na to, by skrypt zawierający polecenia dla ps.output był jednocześnie skryptem powłoki systemowej i by można było go uruchomić bez podawania dodatkowych opcji, mając uruchomioną sesję systemu GRASS. Odwołanie się do samego polecenia ps.output następuje wówczas wewnątrz takiego skryptu, a w działaniu wykorzystywany jest mechanizm potokowania danych.

Przykładowy skrypt dla programu ps.output:

```
#!/bin/sh

ps.output output=psoutput.ps<<EOF
paper A4
   left 0mm
   right 0mm
   bottom 0mm
   top 0mm
   landscape yes
end

scale 1:125000

maparea
   left 25mm
   top 20mm
   color 255:255:255
   fcolor none
end

grid
   format iho
   fcolor 0:0:0
```

```
   major 10000
          width .2mm
          color 80:80:80
   end
   minor 2000
          width .1mm
          color gray
          style dashed
   end
   subminor 5
end

raster dtm_bystrzyca_m_int
end

vlines poziomice_50_bystrzyca
   type lines
          line
          width 0.3mm
          color 230:180:115
          end
   legend -1
end

vpoints miejscowosci
   label Miasta powyzej 10000
   mieszkancow
   type points
   symbol kartografia/miasto_1
   fcolor white
   line
          width .1mm
          color black
   end
   size 6mm
   where str_1='Bystrzyca Klodzka'
end

vpoints miejscowosci
   label Miejscowosci ponizej 10000
   mieszkancow
   type points
   symbol basic/circle
   fcolor white
   line
          width .1mm
          color black
   end
   size 3mm
   where str_1<>'Bystrzyca Klodzka'

end

draw free
   maplimits
end

draw free
   labels nazwy_miejscowosci_ps
end

draw legend
   rectangle 25mm 150mm 85mm 200mm
   white
   border .1mm
   color black
end

draw free
   north 240mm 35mm -
end

scalebar f
   frame
          where 50mm 170mm
          ref left lower
          border 0mm
          fcolor none
          margin 0
   end
   font
          size 5
          color black
   end
   length 2.5
   height 1mm
   units km km
   major 4 2
   minor 2 2
end

note Wysokosci w m n.p.m.
   frame
          where 30mm 153mm
          ref left lower
          border 0mm
          fcolor none
          margin 0mm
   end
   font
          size 4
          color black
   end
end

note Skala 1:125000 (1 cm - 1,25 km)
   frame
          where 52mm 162.5mm
          ref left lower
          border 0mm
          fcolor none
          margin 0mm
   end
   font
          size 5
          color black
   end
end

note BYSTRZYCA KLODZKA
   frame
          where 65mm 151mm
          ref center upper
          border 0mm
          fcolor none
          margin 0mm
   end
   font
          name Helvetica-Bold
          size 8
          color black
   end
end

note OROGRAFIA GMINY
   frame
          where 65mm 157mm
          ref center upper
          border 0mm
          fcolor none
          margin 0mm
   end
   font
          name Helvetica-Bold
          size 6
          color black
   end
end

rlegend
   quantize yes
   raster dtm_bystrzyca_m
   frame
          where 30mm 155mm
          ref left upper
          offset 0 0
          border 0mm
          fcolor none
```



Rys. 8.6. Mapa uzyskana za pomocą polecenia ps.output i przykładowego skryptu.

```
            margin 0mm
    end
font
            name Helvetica
            size 5
    end
    range 300 1300
    height 35mm
    whiteframe 0
end

vlegend
    frame
            where 65mm 175mm
            ref center upper
            offset 0 0
            border 0mm
            fcolor none
            margin 0mm
    end
    font
            name Helvetica
            size 4
    end
    interline 3mm
end

draw free
    psfile 0.06 62.5mm 187.5mm
    WGUG_warsztaty_logo_new.eps
end

note (C)2010 WGUG - Mapa ilustrujaca
    dzialanie ps.output
    frame
            where 65mm 188mm
            ref center upper
            border 0mm
            fcolor none
            margin 0mm
    end
    font
            name Helvetica
            size 4
            color black
    end
end

note Odwzorowanie konforemne (quasi-
    stereograficzne)|Uklad
    wspolrzednych PUWG 1965, strefa IV
    (EPSG 2174)
    frame
            where 230mm 200mm
            ref left upper
            border 0mm
            fcolor none
            margin 0mm
    end
    font
            name Helvetica
            size 4
            color black
    end
end

note na wschod od Greenwich
    frame
            where 15mm 15mm
            ref left lower
            border 0mm
            fcolor none
            margin 0mm
    end
    font
            name Helvetica
            size 6
            color black
    end
end

EOF
```

W skrypcie wykorzystanym do stworzenia mapy (rys. 8.6) użyto zestawu 12 poleceń modułu ps.output. Ich działanie jest identyczne lub bardzo zbliżone do poleceń ps.map, gdyż celem autora tego dodatku było rozszerzenie funkcjonalności modułu ps.map, a nie tworzenie modułu zupełnie od nowa. Podobnie jak w przypadku ps.map użytkownik napotka tu polecenia blokowe (zamykane klauzulą end) i posiadające jedną opcję. W skrypcie występują polecenia, z którymi użytkownik może spotkać się podczas używania ps.map, jak np. paper (określające format papieru, na którym ma się znaleźć docelowa mapa), scale (skala, w jakiej mapa ma być wyrysowana), raster (wskazanie warstwy rastrowej, którą należy nanieść na mapę). W dość podobny sposób używa się poleceń vlines i vpoints.

To, co odróżnia ps.output od ps.map, to wzbogacenie zestawu poleceń pozwalających na ozdabianie map ramkami (grid), czy formatowanie łańcuchów tekstowych (polecenie note) umieszczanych na mapie, a także legend opisujących warstwy wektorowe (vlegend) i rastrowe naniesione na mapę (rlegend).

Użytkownik ma również większe możliwości wyboru w przypadku nanoszenia podziałki liniowej (scalebar).

Autor dodatku wyposażył również ps.output w bardzo funkcjonalne polecenie draw, dzięki któremu można na mapie umieszczać zarówno symbole, opisy (np. granic obszaru mapy), jak i zewnętrzne pliki eps, etykiety przypisane do warstw, jak również proste elementy geometryczne (np. biały prostokąt stanowiący tło dla legendy). Polecenie to pozwala również, za pomocą doboru odpowiedniej opcji, na ustalenie kolejności wyrysowywania poszczególnych elementów.

8.6. Podsumowanie

System GRASS, w założeniu swoich twórców, nie miał być programem do komputerowej kartografii. Położono nacisk na rozwój jego możliwości, jako narzędzia do analiz przestrzennych i modelowania. Niemniej, funkcje pomocne w komputerowym tworzeniu map były w systemie obecne i z biegiem czasu zostały rozwinięte. Obecnie użytkownik ma do dyspozycji narzędzia pozwalające na profesjonalne przygotowanie map zarówno w wersji cyfrowej, jak i po przetworzeniu, w postaci papierowej. Mając do dyspozycji system GRASS wraz z zainstalowanym dodatkiem ps.output służącym do generowania map, a także zawartym w standardowej instalacji poleceniem ps.map, w połączeniu z aplikacjami zewnętrznymi, narzędzie dysponujące funkcjami wspomagającymi przygotowanie map spełniających wymagania poprawnej prezentacji kartograficznej.

GRASS dzięki możliwości zapisu map w formatach pozwalających na umieszczenie ich w prezentacjach multimedialnych, jak i na stronach www może być wykorzystany w szybkiej wizualizacji danych przestrzennych w codziennej pracy. Jeśli użytkownikom zależy na przygotowaniu wydruku papierowego, pomocne mogą okazać się: wbudowany w system moduł ps.map, wciąż rozwijany dodatek ps.output. Oba moduły pozwalają na wygenerowanie pliku w formacie PostScript. Format ten jest zrozumiały dla większości urządzeń drukarskich. Wykorzystanie przy tworzeniu map formatu wektorowego PostScript pozwala na wydruk w dostępnych zdefiniowanych formatach (do wielkości A0 włącznie), lub też w dowolnie zdefiniowanej przez użytkownika wielkości, ze stałą wysoką jakością.

Wygenerowanie plików w formacie PostScript ma kolejną zaletę z punktu widzenia edytora mapy. Pliki w tym formacie można wczytać do zewnętrznych aplikacji obsługujących formaty wektorowe np. Inkscape, Corel Draw, czy Adobe Illustrator. W tych programach można edytować i obrabiać projekty map przed ostatecznym zatwierdzeniem publikacji do druku.

Wykorzystując różne opcje prezentacji kartograficznej systemu GRASS uzyskujemy narzędzia zarówno do szybkiego przygotowania prostych przedstawień kartograficznych jak i do zaawansowanej produkcji map.

Wyniki otrzymane w ten sposób są poprawne kartograficznie, a także nie ustępują produktom własnościowym.

Literatura

[1] Kraak M., Ormeling F., 1998, *Kartografia, wizualizacja danych przestrzennych*, PWN, Warszawa, ss. 275
[2] Litwin L., Myrda G., 2005, *Systemy Informacji Geograficznej – Zarządzanie danymi przestrzennymi w GIS, SIP, SIT, LIS*, Helion, Gliwice, ss.286
[3] Pasławski J. (red.), 2006, *Wprowadzenie do kartografii i topografii*, Nowa Era, Wrocław, ss. 337
[4] Perring F. H., Walters S. M. (edyt.), 1962, *Atlas of the British Flora*, Thomas Nelson & Sons, London, s. 456

Źródła internetowe

[5] Strona internetowa aplikacji Inkscape, http://www.inkscape.org
[6] Strona internetowa aplikacji GSView, http://pages.cs.wisc.edu/~ghost/gsview
[7] Przykłady użycia polecenia ps.map, http://grass.osgeo.org/wiki/ps.map_scripts
[8] Przykłady użycia polecania ps.output/ps.out, http://grass.osgeo.org/wiki/ps.out
[9] Przykładowe realizacje map przy użyciu ps.map, http://geog-pc40.ulb.ac.be/grass/psmap
[10] Opis modułu ps.output/ps.out, https://trac.osgeo.org/grass/browser/grass-addons/postscript/ps.output/ description.html
[11] Strony pomocy dla programu ArcGIS, http://webhelp.esri.com/arcgisdesktop/9.3/

9. Implementacja sztucznych sieci neuronowych w systemie GRASS

Paweł Netzel

9.1. Wprowadzenie

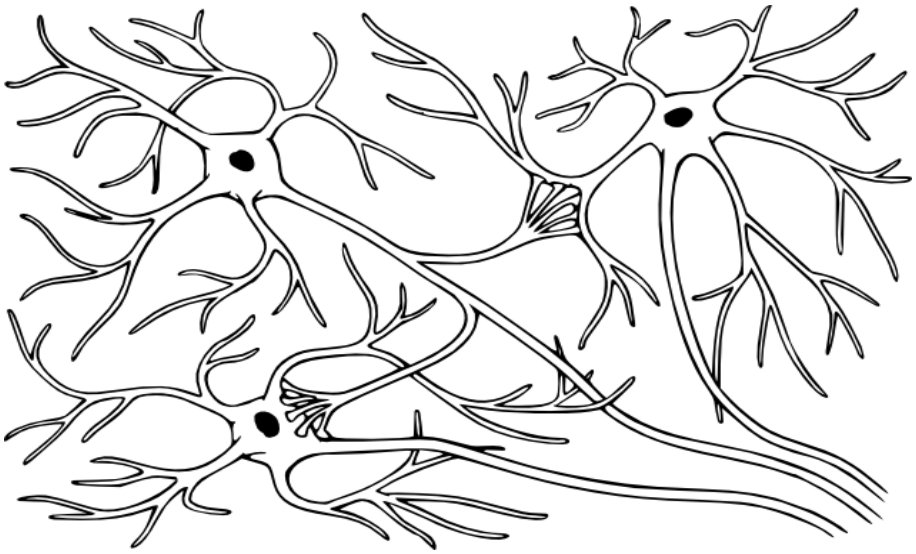
Jednym z ważnych problemów w przetwarzaniu danych przestrzennych jest interpolacja przestrzenna, tworzenie danych gridowych na podstawie danych punktowych (lub liniowych). Istnieje wiele algorytmów pozwalających na interpolację przestrzenną. Począwszy od prostych średnich ważonych, poprzez funkcje sklejalne, aż po metody geostatystyki takie jak kriging lub cokiging. Każda z wymienionych metod obarczona jest błędem interpolacji oraz każda z nich wprowadza zniekształcenia związane z aparatem matematycznym wykorzystanym do interpolacji.

Temat poszukiwania metod interpolacji przestrzennej, jak również modelowania rozkładów przestrzennych, jest tematem otwartym. Wprowadzane są i testowane coraz to nowsze metody.

Sztuczne sieci neuronowe mogą zostać również wykorzystane do modelowanie i interpolacji przestrzennej. Prace nad wykorzystaniem sztucznych sieci neuronowych do tego celu są podejmowane w różnych miejscach na świecie. Sieci neuronowe wykorzystywane są zarówno do budowania modeli zjawisk jak i modeli przestrzennych [1], jak też do interpolowania rozkładów przestrzennych na podstawie danych punktowych [3, 7]. Porównanie wyników otrzymanych za pomocą sieci z wynikami stosowanych obecnie metod są bardzo obiecujące [3].

Aby wykorzystać sztuczne sieci neuronowe do interpolacji przestrzennej w systemie GIS konieczne jest zaimplementowanie algorytmów związanych z sieciami neuronowymi. Takie implementacje istnieją w rozwiązaniach własnościowych (np. ArcGIS „Neural Network Extension").

System GRASS jest rozwiązaniem FOSS (ang. *Free Open Souce Software*) dostępnym na licencji GNU GPL. Ma on architekturę otwartą i modułową. Składa się on ze zbioru poleceń uruchamianych z linii komend. Dzięki takiej architekturze łatwo go rozszerzać poprzez dodawania nowych poleceń tworzonych w języku C lub w języku skryptowym (język powłoki systemu operacyjnego, Python). Pozwala to na zaimplementowanie nowych rozwiązań takich jak moduły obsługi sieci neuronowych. Dostępne interfejsy programistyczne API (ang. *Application Programming Interface*), w szczególności dla języka Python, umożliwiają rozszerzanie systemu GRASS o dostęp do bibliotek zewnętrznych i wykorzystanie pełni ich funkcjonalności. Wśród takich bibliotek zawierających algorytmy przetwarzania danych są też biblioteki realizujące sztuczne sieci neuronowe. Niektóre z tych bibliotek dostępne są na licencji pozwalającej na wykorzystanie ich w systemach FOSS. Biblioteką, która spełnia takie założenia jest FANN (ang. *Fast Artificial Neural Network*). Biblioteka FANN dostępna jest na licencji LGPL (ang. *Lesser Gnu Public Licence*), posiada API, między innymi dla języka Python, i realizuje obliczenia wykorzystując sztuczne sieci neuronowe [4, 9].



Rys. 9.1 Przykład sieci złożonej z trzech neuronów biologicznych

9.2. Sztuczne sieci neuronowe

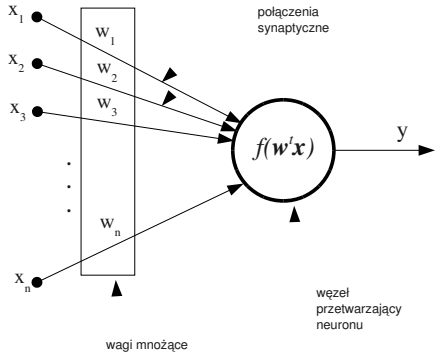
9.2.1. Neuron

Neuron biologiczny jest to komórka posiadająca zdolność wykonywania pewnego przetwarzania danych. Jest on pobudzany przez jedno lub wiele wejść, a sygnał wyjściowy (reakcja komórki na bodźce wejściowe) przekazuje do jednego lub wielu neuronów połączonych z nim. W neuronie biologicznym (rys. 9.1) sygnał wyjściowy zależy od wartości wejściowych w sposób szalenie złożony jako efekt wewnątrzkomórkowych przemian elektrochemicznych.

Jednak proste modele przybliżające zasadę działania neuronu biologicznego, które uwzględniają tylko najbardziej podstawowe procesy, prowadzą do działających rozwiązań problemów praktycznych.

Sztuczny neuron, analogicznie do neuronu biologicznego ma wiele wejść. Sygnał z każdego wejścia jest modyfikowany, przemnażany przez odpowiedni współczynnik nazywany wagą. Następnie tak zmodyfikowane sygnały są sumowane, a suma traktowana jest jako wartość wejściowa nieliniowej funkcji – funkcji aktywacji neuronu.

Wartość tej funkcji jest wielkością wyjściową pobudzenia neuronu.



Rys. 9.2 Ogólny model sztucznego neuronu

Sposób i zakres reakcji sztucznego neuronu definiowany jest przez określenie wektora wag oraz wybór funkcji aktywacji.

Ogólny, uproszczony model matematyczny sztucznego neuronu można zilustrować w sposób przedstawiony na rysunku 9.2 [8].

Ujmując to matematycznie:

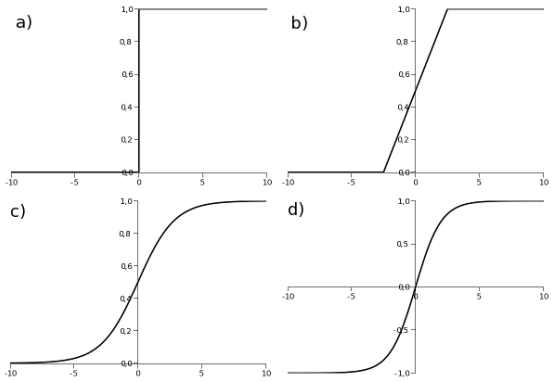
$$y = f(\boldsymbol{w}^t \boldsymbol{x}) = f(\sum_{i=1}^{n} w_i x_i)$$

gdzie

y - odpowiedź neuronu,

$\boldsymbol{w} = [w_1, w_2, \ldots, w_n]^t$ - wektor wag,

$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^t$ - wektor wejściowy.



Rys. 9.3 Wykresy przykładowych funkcji aktywacji neuronu

Wektory są definiowane jako wektory kolumnowe, a indeks t oznacza transpozycję. Funkcja $f(\boldsymbol{w}^t\boldsymbol{x})$ nazywana jest funkcją przejścia lub funkcją aktywacji.

Jako funkcję aktywacji często stosuje się jedną z następujących funkcji:

funkcja skoku (rys. 9.3a)

$$f(x)=\begin{cases}0, & dla\ x<0\\ 1, & dla\ x\geqslant 0\end{cases}$$

funkcja liniowa ograniczona (rys. 9.3b)

$$f(x)=\begin{cases}0, & dla\ x<-\dfrac{1}{2a}\\ ax+\dfrac{1}{2}, & dla -\dfrac{1}{2a}\leqslant x<\dfrac{1}{2a}\\ 1, & dla\ x\geqslant\dfrac{1}{2a}\end{cases}$$

funkcja logistyczna (rys. 9.3c)

$$f(x)=\frac{1}{1+e^{-ax}}$$

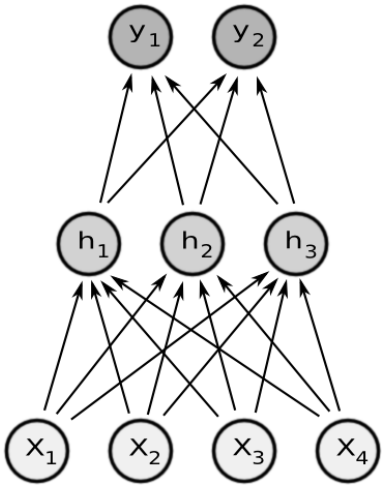
tangens hiperboliczny (rys. 9.3d)

$$f(x)=\tanh(ax)$$

Oczywiście nie wyczerpuje to zbioru funkcji aktywacji stosowanych w sztucznych sieciach neuronowych.

9.2.2. Sieć neuronowa

Sztuczne neurony można łączyć ze sobą na różne sposoby. Każdy typ sieci jest ściśle powiązany z odpowiednią metodą doboru wag czyli metodą uczenia sieci. Typ sieci łączy się też z pewnymi grupami problemów, w których rozwiązaniu dobrze się sprawdza.



Rys. 9.4 Jednokierunkowa sieć warstwowa z jedną warstwą ukrytą. x_i – neurony wejściowe, y_i – neurony wyjściowe, h_i – neurony warstwy ukrytej

Wśród tej wielości różnych architektur sztucznych sieci neuronowych istnieją modele, których wielka wartość praktyczna została sprawdzona. Pomimo olbrzymich uproszczeń wprowadzonych w nich w stosunku do oryginału biologicznego, zachowują one umiejętność generalizacji posiadanej wiedzy.

Taką sprawdzoną architekturą sztucznej sieci neuronowej jest jednokierunkowa sieć warstwowa - perceptron wielowarstwowy.

Sieć taka składa się z neuronów ułożonych w warstwy. Wyróżniana jest warstwa wejściowa, warstwa wyjściowa oraz jedna (lub więcej) warstwa ukryta. Każdy neuron warstwy poprzedniej może łączyć się ze wszystkimi neuronami warstwy następnej. Sieć o takiej topologii przedstawiono na rysunku (rys. 9.4). Składa się ona z 4 neuronów w warstwie wejściowej, jednej warstwy ukrytej złożonej z trzech neuronów oraz ma dwa neurony w warstwie wyjściowej.

Tego typu sieci sprawdzają się dobrze w zakresie aproksymacji funkcji, prognozowania klasyfikacji i rozpoznawania wzorców [5]. Ponadto algorytm uczenia takiej sieci – algorytm wstecznej propagacji błędu (ang. *backpropagation algorythm*) – jest dobrze przebadany i udokumentowany.

9.2.3. Algorytm uczenia sieci

Uczenie sieci algorytmem wstecznej propagacji błędu jest dobrze opisane [5, 6, 8]. W podanej literaturze można znaleźć zarówno opisy samej idei tego algorytmu, zapis matematyczny, jak i kompletne implementacje w językach programowania.

Proces uczenia zaczyna się od zainicjowania wag wszystkich neuronów. Przyjmują one małe, niezerowe, losowo dobrane wartości. Kolejnym krokiem jest podanie na wejście sieci danych ze zbioru uczącego i obliczenie odpowiedzi sieci. Następnie obliczana jest miara błędu sieci.

W oparciu o wygenerowany błąd sieci, modyfikowane są wagi neuronów w celu jego zmniejszenia. Proces ten powtarzany jest wielokrotnie, aż do skutku [5] czyli osiągnięcia kryterium, które przyjęto dla określenia czy sieć jest nauczona.

Cały cykl przejścia:

- prezentacja próbek,
- pomiar błędu,
- modyfikacja wag

nazywany jest epoką.

Przyjmijmy sieć o *n* neuronach w warstwie wyjściowej i załóżmy, że przetwarzamy wzorzec uczący o numerze *p*. Niech t_{pj} oznacza prawidłową (oczekiwaną) aktywację *j*-tego neuronu wyjściowego a y_{pj} wartość wygenerowaną przez neuron. Wtedy błąd jednej prezentacji wynosi:

$$E_p = \frac{1}{n}\sum_{j=0}^{n-1}(t_{pj}-y_{pj})^2$$

a przy założeniu, że epoka liczy *m* prezentacji, błąd epoki wynosi:

$$E = \frac{1}{m}\sum_{p=0}^{m-1} E_p .$$

Wtedy, dla jednej prezentacji pochodna błędu względem wagi łączącej *j*-ty neuron wyjściowy z *i*-tym neuronem w warstwie poprzedzającej wynosi

$$\frac{\partial E}{\partial w_{ij}} = -y_i f'(\sum_{i=1}^{n} w_{ij}x_{ij})(t_j - y_j).$$

Można to zapisać w następujący sposób

$$\delta_j = f'(\sum_{i=1}^{n} w_{ij}x_{ij})(t_j - y_j),$$

$$\frac{\partial E}{\partial w_{ij}} = -y_i \delta_j .$$

Dla dowolnej wagi łączącej *j*-ty neuron w warstwie ukrytej z *i*-tym neuronem warstwy poprzedniej pochodna $\frac{\partial E}{\partial w_{ij}}$ wynosi

$$\delta_j = f'(\sum_{i=1}^{n} w_{ij}x_{ij})\sum_{k}(\delta_k w_{kj}),$$

$$\frac{\partial E}{\partial w_{ji}} = -y_j \delta_j$$

gdzie

w_{kj} - waga łącząca *j*-ty neuron w warstwie ukrytej z *k*-tym neuronem w warstwie następnej.

W ten sposób na podstawie wartości δ z warstwy następnej można obliczyć pochodne w warstwie wstecznej (stąd też nazwa algorytmu – algorytm wstecznej propagacji błędu).

Uwzględniając modyfikację związaną z wprowadzeniem parametru α w celu przyspieszenia procesu uczenia (parametr ten nosi nazwę momentum), wagi modyfikowane są zgodnie ze wzorem:

$$w_{jik} = w_{ji(k-1)} + \eta\delta_{jk}y_{jk} + \alpha\eta\delta_{j(k-1)}y_{j(k-1)}.$$

Parametr η odpowiada za prędkość uczenia się, a k oznacza numer prezentacji wzorca.

W celu uniknięcia wpadania stanu wag sieci w oscylacje wynikające z powtarzającej się kolejności próbek zbioru uczącego, kolejność ta jest zmieniana w sposób losowy w każdej epoce.

9.3. Istniejące implementacje sieci neuronowych dostępne dla GRASS

System GRASS miał kiedyś moduł realizujący obliczenia w oparciu o sieci neuronowe. Moduł ten istniał na przełomie wersji 4 i 5. Potem wykorzystywano przede wszystkim zewnętrzne implementacje sieci neuronowych. Przykładem może być SNNS (ang. *Stuttgart Neural Network Symulator*). Wymagało to transmisji danych do oraz z systemu GRASS w formacie tekstowym. Innym sposobem na uruchomienie obliczeń neuronowych w systemie GRASS jest wykorzystanie pakietów takich jak R lub Octave oraz bibliotek łączących je z systemem GRASS.

Za każdym jednak razem niezbędne jest konwertowanie danych z jednych struktur (wewnętrznych systemu GIS) na inne struktury (wewnętrzne systemu R lub Octave). Zaletą tego rozwiązania jest możliwość użycia wielu różnych topologii sieci oraz różnych algorytmów uczenia, które są zaimplementowane na przykład w systemie R. Przeniesienie danych do systemu R realizowane jest przez dedykowane funkcje łączące R z systemem GRASS. Problemem jest jednak to, że struktury danych, które są przetwarzane, umieszczane są w całości w pamięci operacyjnej komputera. Powoduje to poważne problemy przy obliczeniach wymagających dostępu do wielu dużych warstw rastrowych.

Najzręczniejszym wyjściem omijającym powyższy problem jest uruchomienie obliczeń bezpośrednio na danych z systemu GIS.

9.4. GRASS – Python – FANN

Jest wiele bibliotek implementujących różne architektury sztucznych sieci neuronowych. Jedną z nich jest FANN (ang. *Fast Artificial Neural Network*). Biblioteka FANN dostępna jest na licencji LGPL i pozwala na realizację jednokierunkowych sieci neuronowych uczonych algorytmem wstecznej propagacji błędu [9].

Wspólną platformą, na której można połączyć system GRASS z biblioteką FANN, są skrypty systemu GRASS realizowane w języku Python. Przykładem takich skryptów mogą być przedstawione dalej skrypty stworzone przez autora.

GRASS posiada interfejs programistyczny API dający dostęp zarówno do wywoływania gotowych poleceń GRASS, jak i pozwalający na przetwarzanie danych GIS na poziomie elementów rastra i obiektów wektorowych. Biblioteka FANN ma także API w języku Python [10].

Platforma języka Python pozwala więc na budowę pomostu pomiędzy GRASS i FANN i w rezultacie implementację sieci neuronowych bezpośrednio w systemie GRASS.

9.5. Moduły ann.*

Moduły tworzące pakiet pozwalający na pracę z sieciami neuronowymi oznaczono przedrostkiem „ann”. Ponieważ pakiet ann związany jest z zarządzaniem definicjami sieci, uczeniem sieci, pozyskiwaniem danych do sieci oraz tworzeniem nowych warstw z wykorzystaniem sieci neuronowych i ma docelowo pracować zarówno z danymi rastrowymi, jak i wektorowymi nie przypisano modułów do konkretnej grupy r.* lub v.* a jedynie wykorzystano przedrostek ann (od ang. *Artificial Neural Network*).

W chwili obecnej zaimplementowane moduły tworzą warstwy rastrowe i pozwalaj przede wszystkim na interpolację danych przestrzennych.

Zestaw poleceń pozwalających na obsługę sieci neuronowych:

- ann.create – tworzenie sieci,
- ann.info – informacja o sieci,
- ann.data.rast – pobranie danych uczących,
- ann.data.vect – pobieranie danych uczących (planowany moduł),
- ann.learn – uczenie sieci,
- ann.run.rast – uruchomienie sieci,
- ann.run.vect – uruchamianie sieci (planowany moduł).

Poniżej zamieszczony jest krótki opis działających poleceń.

9.5.1. Polecenie ann.create

Polecenie ann.create tworzy sieć o zadanej architekturze. Składnia wywołania polecenia:

```
ann.create [-l] in=value
   hidd=value[,value,...] out=value
   net=filename [conn_rate=value]
   [learn_rate=value] [--verbose] [--
   quiet]
```

Opcje:

-l ustawia liniową funkcję aktywacji dla neuronów wyjściowych.

Parametry:

in liczba neuronów wejściowych (domyślnie: 1),

hidd liczby neuronów w warstwach ukrytych (domyślnie: jedna warstwa z 2 neuronami),

out liczba neuronów wyjściowych (domyślnie: 1),

net nazwa pliku z siecią xxxx.net

conn_rate połączenia (0.0 – brak, 1.0 – pełne) (domyślnie: 1),

learn_rate szybkość uczenia (domyślnie: 0.7).

Parametr conn_rate określa jak wiele połączeń zostanie utworzonych w sieci. Gdy conn_rate zostanie nadana wielkość 1, w sieci zostaną utworzone wszystkie połączenia pomiędzy neuronami (ang. *fully connected network*). W przypadku nadani wielkości 0.5 tylko połowa połączeń pomiędzy neuronami sieci zostanie utworzona.

W wyniku powstaje plik tekstowy zawierający definicje sieci. Plik tworzono jest w bieżącym katalogu użytkownika. Plik ma automatycznie dodawane rozszerzenie „.net”.

9.5.2. Polecenie ann.info

Wypisuje dostępne sieci neuronowe lub opisuje wybraną sieć. Informacje o sieciach pobierane są z plików *.net zawierających definicje sieci. Pod uwagę brane są pliki znajdujące się w bieżącym katalogu. Składnia wywołania polecenia jest następująca:

```
ann.info [-l] [net=filename]
   [--verbose] [--quiet]
```

Opcje:

-l wypisuje wszystkie definicje sieci neuronowych zawarte w bieżącej kartotece.

Parametry:

net nazwa wybranej sieci (zostanie wypisana informacja o tej sieci).

9.5.3. Polecenie ann.data.rast

Przygotowuje dane uczące na podstawie lokalizacji oraz warstw rastrowych. Tworzony jest plik z danymi o zadanej nazwie będący wynikiem próbkowania warstw rastrowych. Plik budowany jest na podstawie próbkowania warstw rastrowych w zadanych lokalizacjach. Lokalizacje punktów można wskazać podając ich koordynaty w poleceniu, wskazując plik tekstowy z koordynatami lub podając nazwę warstwy wektorowej z punktami.

Składnia wywołania programu wygląda następująco:

```
ann.data.rast in=string[,string,...]
   out=string[,string,...]
   [vector=string] [points=filename]
   [east_north=string]
   output=filename
```

Parametry:

in lista warstw rastrowych, z których zostaną pobrane dane wejściowe,

out lista warstw rastrowych, z których zostaną pobrane dane wyjściowe,

vector warstwa wektorowa zawierająca punkty wskazujące lokalizacje danych uczących,

points plik tekstowy zawierający współrzędne punktów,

east_north lista współrzędnych punktów,

output nazwa pliku, do którego zostanie zapisany zbiór danych uczących.

9.5.4. Polecenie ann.learn

Polecenie to uruchamia proces uczenia sieci neuronowej. Uczenie odbywa się w oparciu o zadany, przygotowany wcześniej, zbiór uczący. Wynik pracy programu, nauczona sieć neuronowa, zapisywany jest pod zadaną nazwą. Jeżeli nie wskazano nazwy

nowego pliku, to nadpisywana jest sieć podana jako wejściowa. Składnia wywołania programu:

```
ann.learn net=filename data=filename
   [max_iter=value]
   [iter_report=value] [error=value]
   [output=string]
```

Parametry:

net nazwa sieci (pliku z siecią),

data nazwa pliku ze zbiorem uczącym,

max_iter maksymalna liczba epok,

iter_report liczba epok, po której wypisywany jest raport,

error dopuszczalny błąd,

output nazwa pliku, do którego zostanie zapisana nauczona sieć.

9.5.5. Polecenie ann.run.rast

Polecenie generuje wynikowe warstwy korzystając z nauczonej sieci.

Składnia wywołania polecenia:

```
ann.run.rast in=string[,string,...]
   [net=filename]
   output=string[,string,...]
   [--verbose] [--quiet]
```

Parametry:

in warstwy rastrowe z danymi wejściowymi,

net nazwa sieci,

output warstwy rastrowe wygenerowane przez sieć.

9.6. Dodatkowe oprogramowanie wykorzystujące FANN

Poza wykorzystaniem modułów zaimplementowanych w systemie GRASS można użyć narzędzi graficznych wykorzystujących bibliotekę FANN oraz jej formaty danych. Mogą być one wykorzystane w systemie GRASS jedynie do dwóch etapów: do tworzenia oraz do uczenia sieci neuronowej.

Narzędzia te wyposażone są w graficzny interfejs użytkownika. Przykładem takiego programu jest FannTool autorstwa Derina Deli Mavi. Narzędzie to było testowane w wersji 1.1.

FannTool udostępnia możliwość pracy z wykorzystaniem interfejsu graficznego. Można z jego wykorzystaniem:

- przygotować dane,
- stworzyć sieć,
- wczytać dane uczące,
- przeprowadzić proces uczenia sieci,
- wczytać dane testowe,
- przetestować nauczoną sieć,
- uruchomić sieć dla zbioru danych wejściowych.

Narzędzie jest bardzo poręczne, lecz jest ono przeznaczone do prac niezależnie od systemu GRASS. W związku z tym, z całej dostarczanej funkcjonalności, użyteczne jest jedynie tworzenie, uczenie i testowanie sieci neuronowej.

Przygotowanie danych, jak również uruchomienie sieci na danych z systemu GRASS realizowane jest efektywniej z wykorzystaniem modułów ann.*.

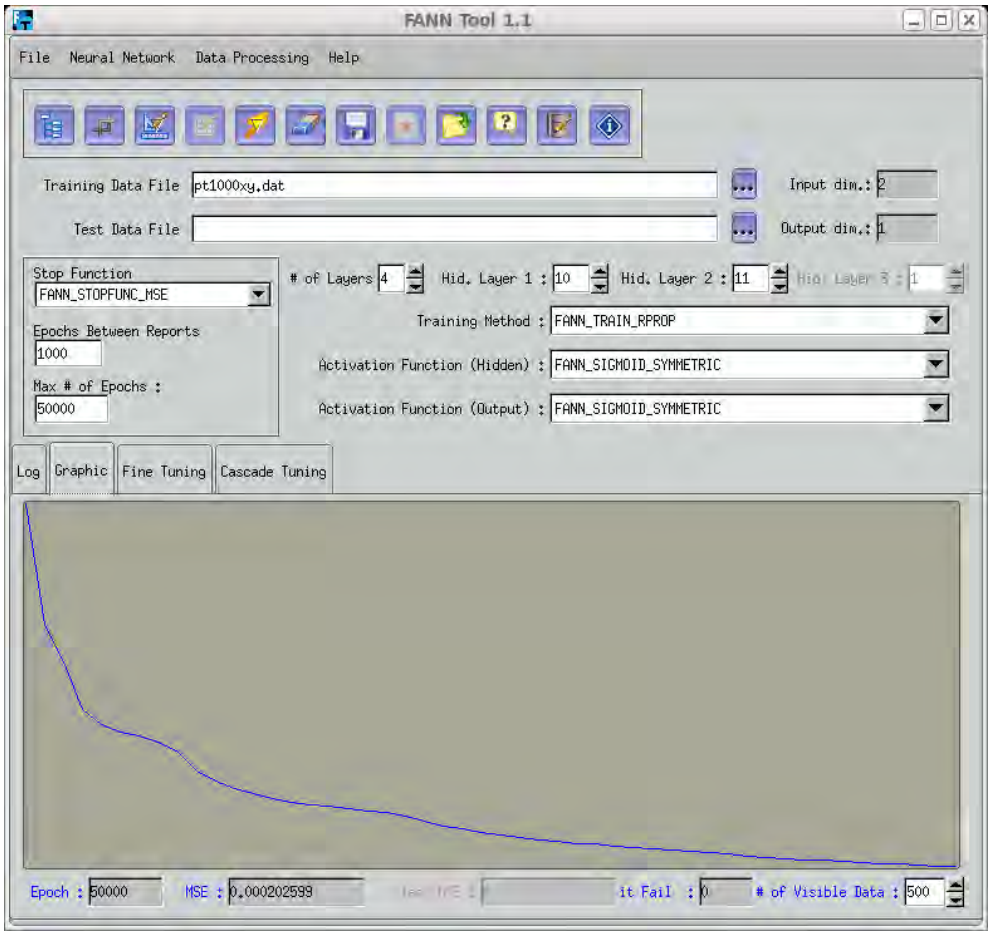
Wygląd interfejsu programu FannTool przedstawiono na rysunku 9.5.

9.7. Przykład wykorzystania sieci neuronowych do interpolacji cyfrowego modelu terenu

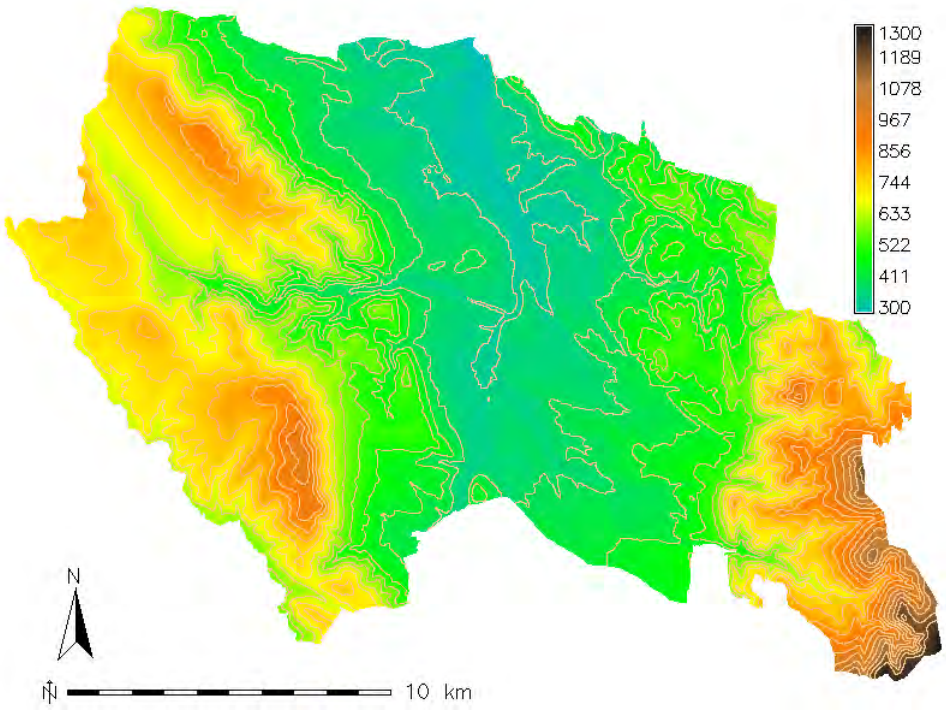
Pakiet poleceń ann.* wykorzystano do interpolacji modelu terenu na podstawie losowo rozproszonych punktów wysokościowych, aby przetestować jego działanie oraz przydatność. Do testów przyjęto model terenu dla gminy Bystrzyca Kłodzka w układzie współrzędnych PUWG 65 strefa IV (EPSG 2174). Zasięg przestrzenny modelu (rys. 9.6) to 22km x 29km (2726 x 2068 elementów rastra). Wielkość elementu rastra to ok. 10.64 x 10.64 m. Model terenu obejmuje teren zawierający się w granicach administracyjnych gminy. Rozdzielczość pionowa modelu to 0.1 m.

Dla tak zdefiniowanego obszaru wylosowano trzy zbiory punktów odpowiednio 1000, 5000 i 10000 punktów. Następnie z wylosowanych zbiorów wybrano jedynie te punkty, które znajdowały się w granicach administracyjnych gminy. W wyniku otrzymano 3 zbiory punktów, losowo rozrzuconych na terenie gminy Bystrzyca Kłodzka, o liczebnościach:

- zbiór 1 – 532 punkty
- zbiór 2 – 2654 punkty
- zbiór 3 – 5314 punktów



Rys. 9.5 Interfejs użytkownika programu FannTool



Rys. 9.6 Oryginalny numeryczny model terenu, dla którego przeprowadzono interpolację - obręb gminy Bystrzyca Kłodzka

Zbiory punktów, dla których przeprowadzono testy zilustrowano na rysunku 9.7.

Wykorzystując utworzone w ten sposób zbiory punktów uczących przygotowano, za pomocą polecenia ann.data.rast, zbiory uczące. Jako dane wejściowe przyjęto współrzędne X, Y punktu a jako dane wyjściowe – wysokość nad poziom morza w metrach.

Dane uczące zostały następnie unormowane tak, aby każdy z parametrów zawierał się w przedziale (0.05, 0.95). Otrzymane w ten sposób dane wykorzystano do uczenia sieci.

Architekturę sieci określono empirycznie dobierając liczbę neuronów tak, aby sieć była w stanie nauczyć się zbioru uczącego z błędem nie większym niż 0.001 w czasie krótszym niż 100000 epok. W rezultacie otrzymano następujące sieci przedstawione w tabeli 9.1. W kolumnie „Liczba neuronów ukrytych”, po przecinkach, podano liczebności neuronów w kolejnych warstwach ukrytych.

Sieci były tworzone z pomocą polecenia ann.create i uczone przy użyciu polecenia ann.learn. Otrzymane sieci posłużyły następnie do wygenerowania modelu terenu. Warstwy wynikowe zawierały wielkości odpowiadające przeskalowanym wynikom. Aby otrzymać rzeczywiste wysokości w metrach wyniki ponownie przeskalowano.

Model wysokościowy otrzymany przy użyciu sieci 3 przedstawiono na rysunkach 9.8. Dodatkowo naniesiono poziomice z oryginalnego modelu terenu. Poziomice wygenerowane zostały co 50m w pionie.

Poniżej przedstawiona jest lista poleceń realizujących: przygotowanie danych, tworzenie sieci, uczenie sieci oraz generowanie wyników:

- przygotowanie danych

```
r.mapcalc 'x=(x()-3686000)/(3715000-
   3686000)'
r.mapcalc 'y=(y()-5462000)/(5484000-
   5462000)'
r.mapcalc 'dtmm=dtm_bystrzyca_m/
   1300.0'
v.random -d output=pts5000 n=5000
   column="v DOUBLE PRECISION"
v.what.rast raster=dtm_bystrzyca_m
   layer=1 vector=pts5000 column=v
```

- utworzenie zbioru uczącego

```
ann.data.rast in=x,y out=dtmm
   output=pt5000xy vector=pts5000a
```

- zdefiniowanie sieci

```
ann.create -l in=2 hidd=15,9,5 out=1
   net=dtm_5000xy
```

- uczenie sieci

```
ann.learn net=dtm_5000xy
   data=pt5000xy output=dtm_5000xy_l
   max_iter=100000 error=0.001
```

- wygenerowanie warstwy wysokości

```
ann.run.rast in=x,y net=dtm_5000xy.l
   output=dtm5000xy
r.mapcalc 'dtm_nn_5000xy_real=1300.0*
   dtm_nn_5000xy'
```

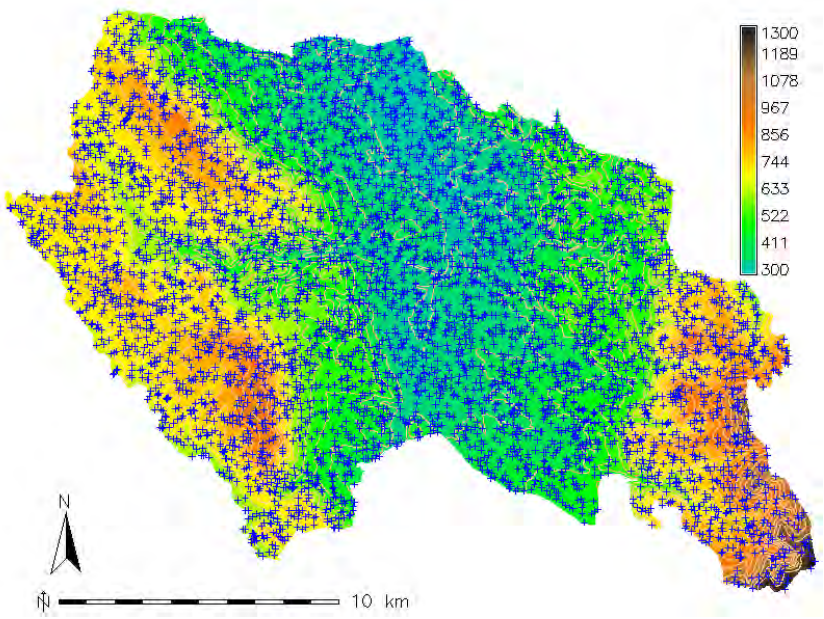
9.8. Porównanie wyników z metodami interpolacji systemu GRASS

Wyniki otrzymane przy pomocy sztucznych sieci neuronowych (ANN) porównano z wynikami otrzymanymi za pomocą algorytmów interpolacji, które są obecnie zaimplementowane w systemie GRASS. Pod uwagę wzięto następujące algorytmy interpolacyjne:

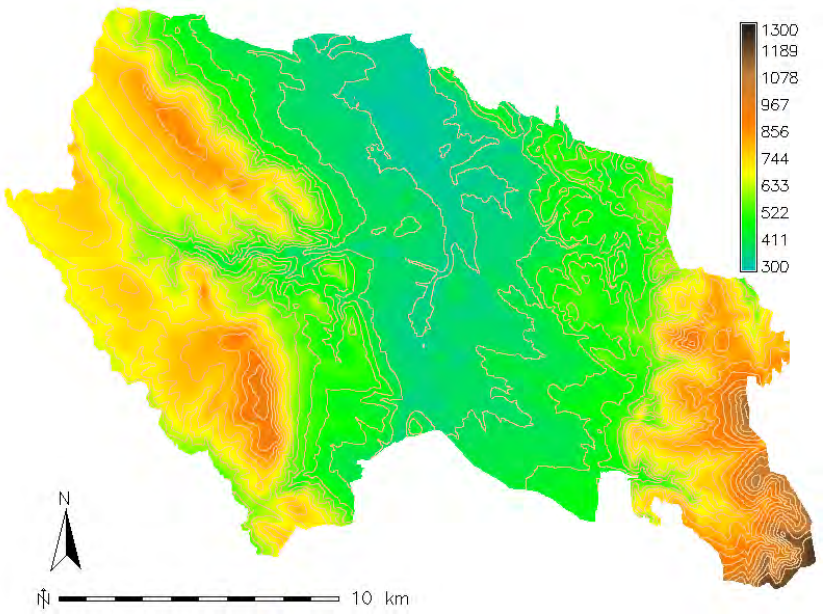
- średnia ważona wykorzystująca odwrotność odległości (IDW1) - polecenie v.surf.idw,
- średnia ważona wykorzystująca kwadrat odwrotności odległości (IDW2) – polecenie v.surf.idw,
- funkcje sklejalne, interpolacja dwuliniowa (BS_Blin) – polecenie v.surf.bspline,
- funkcje sklejalne, interpolacja dwusześcienna (BS_Bcub) – polecenie v.surf.bspline,
- algorytm RST (Regularized Splines with Tension) – polecenie v.surf.rst.

Interpolacje przeprowadzano w oparciu o te same zbiory punktów, które zostały wykorzystane do uczenia sieci neuronowych. Wynikowe modele terenu odniesiono do oryginalnego wysokościowego modelu terenu gminy Bystrzyca Kłodzka.

W tabelach 9.2, 9.3 i 9.4 przedstawiono statystyki różnic pomiędzy modelem wejściowym a modelem interpolowanym dla każdej z testowanych metod interpolacji oraz w rozbiciu według zbiorów danych wejściowych.



Rys. 9.7 Rozkład przestrzenny zbiorów punktów uczących – zbiór 3



Rys. 9.8 Model terenu wygenerowany przez sieć 3 uczoną na podstawie 5314 losowo rozproszonych punktów wysokościowych.

Tabela 9.1 Architektura sieci interpolujących wysokość nad poziom morza dla przygotowanych zbiorów uczących.

Nazwa sieci	Liczba warstw	Liczba neuronów			Liczba epok	Błąd
		wejściowych	ukrytych	wyjściowych		
Sieć 1	4	2	15,11	1	65324	0.00098
Sieć 2	5	2	15,9,5	1	45335	0.00097
Sieć 3	6	2	32,38,26,11	1	42014	0.00099

Tabela 9.2 Statystyki różnic pomiędzy modelem wejściowym a modelem interpolowanym dla różnych metod interpolacji – interpolacja na zbiorze 532 elementów

Nazwa	Średnia	Odchylenie standardowe	Minimum	Maksimum	1 kwartyl	3 kwartyl
IDW1	−21.46	159.02	−319.04	413.66	−162.32	122.49
IDW2	−57.08	175.39	−460.07	380.50	−171.74	88.41
BS_Blin	6.23	60.60	−152.73	508.79	−29.75	30.15
BS_Bcub	6.55	58.87	−152.66	504.04	−27.96	28.83
RST	1.01	36.60	−203.86	381.86	−12.35	10.63
ANN	14.73	47.81	−325.33	400.94	−12.46	40.75

Tabela 9.3 Statystyki różnic pomiędzy modelem wejściowym a modelem interpolowanym dla różnych metod interpolacji – interpolacja na zbiorze 2654 elementów

Nazwa	Średnia	Odchylenie standardowe	Minimum	Maksimum	1 kwartyl	3 kwartyl
IDW1	−40.58	146.58	−435.01	408.84	−138.54	65.17
IDW2	−37.43	191.14	−688.42	388.79	−104.11	83.64
BS_Blin	−0.23	29.14	−142.19	223.09	−15.19	11.59
BS_Bcub	0.01	28.46	−137.38	213.97	−14.44	11.49
RST	−0.15	14.69	−109.00	116.44	−4.74	4.52
ANN	3.00	36.43	−216.84	207.66	−13.97	22.41

Tabela 9.4 Statystyki różnic pomiędzy modelem wejściowym a modelem interpolowanym dla różnych metod interpolacji – interpolacja na zbiorze 5314 elementów

Nazwa	Średnia	Odchylenie standardowe	Minimum	Maksimum	1 kwartyl	3 kwartyl
IDW1	25.18	121.43	−328.61	486.91	−65.77	93.33
IDW2	27.42	132.49	−503.67	479.44	−47.49	101.96
BS_Blin	−0.04	22.59	−130.73	186.40	−10.62	8.75
BS_Bcub	−0.05	21.86	−100.37	169.05	−10.18	8.34
RST	−0.26	9.96	−101.53	83.07	−2.84	2.60
ANN	−0.10	20.90	−142.54	212.34	−11.63	10.00

W tabelach przedstawiono kolejno: średnią różnicę, odchylenie standardowe różnicy, minimalną i maksymalną wielkość różnicy oraz wielkość pierwszego i trzeciego kwartyla różnicy. Statystyki policzono dla zbioru 2995106 niepustych elementów rastra.

Różnice liczono według wzoru

$$d = DTM_{oryg} - DTM_{interp}$$

Dla wszystkich trzech zbiorów punktów metoda interpolacji wykorzystująca sztuczne sieci neuronowe okazała się znacząco lepsza od interpolacji używającej odwrotności odległości (metody IDW1 i IDW2). W przypadku zbioru 1 (najmniej licznego) model terenu wygenerowany przez sieci miał tendencje do niedoszacowania wysokości w modelu wyjściowym. Dla tego zbioru

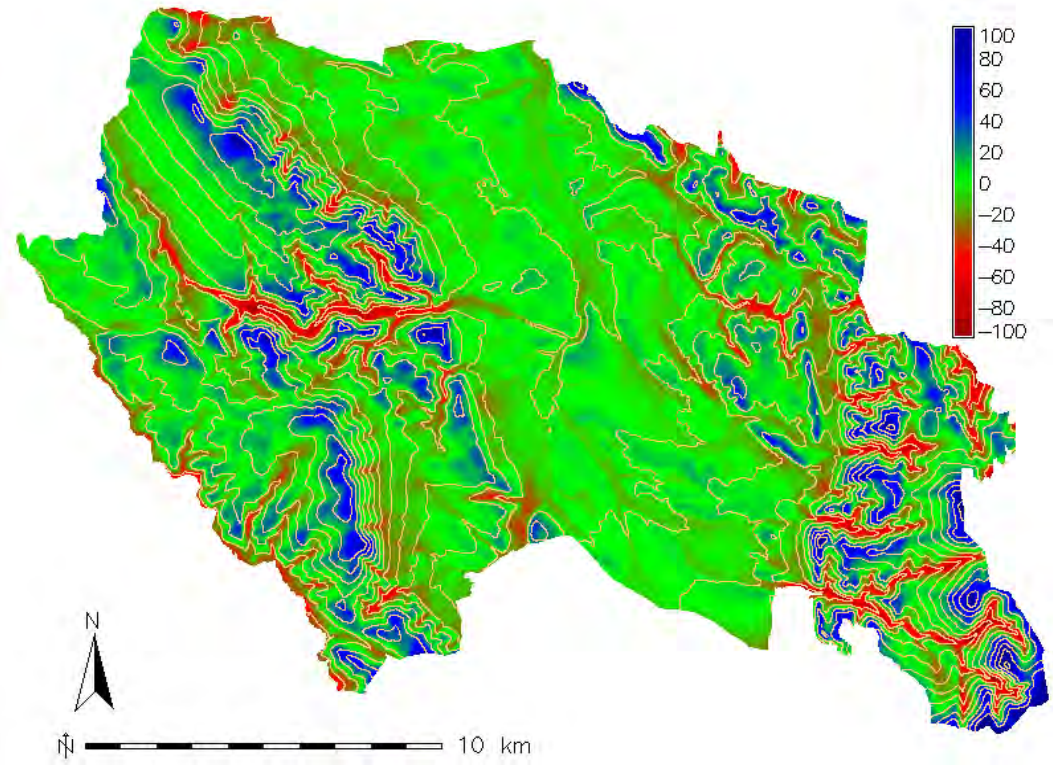
wyniki otrzymane przy pomocy sieci neuronowych były gorsze niż wyniki otrzymane za pomocą funkcji sklejalnych. W przypadku pozostałych dwóch zbiorów statystyczne wyniki interpolacji wykonanej przy pomocy sztucznych sieci neuronowych nie odbiegały od rezultatów otrzymanych za pomocą funkcji sklejalnych. Różnice widać za to w rozkładzie przestrzennym błędów. Interpolacja funkcjami sklejalnymi (BS_Blin i BS_Bcub) wykazują tendencję do spłaszczania interpolowanego modelu. Największe błędy rozkładają się wzdłuż osi dolin oraz wzdłuż grzbietów (rys. 9.9).

Rozkład przestrzenny błędów interpolacji wykonanej przy pomocy sztucznych sieci neuronowych związany jest ze złożonością rzeźby terenu oraz z rozkładem przestrzennym gęstości punktów z których model był interpolowany (rys. 9.10). Nie wykazuje on tak wyraźnych tendencji jak rozkład błędów interpolacji funkcjami sklejalnymi. W przypadku wykorzystani sieci neuronowych, rozkład błędów przypomina rozkład błędów interpolacji za pomocą algorytmu RST. Algorytm RST generuje jednak znacząco mniejszy rozrzut wielkości błędu (rys. 9.11).

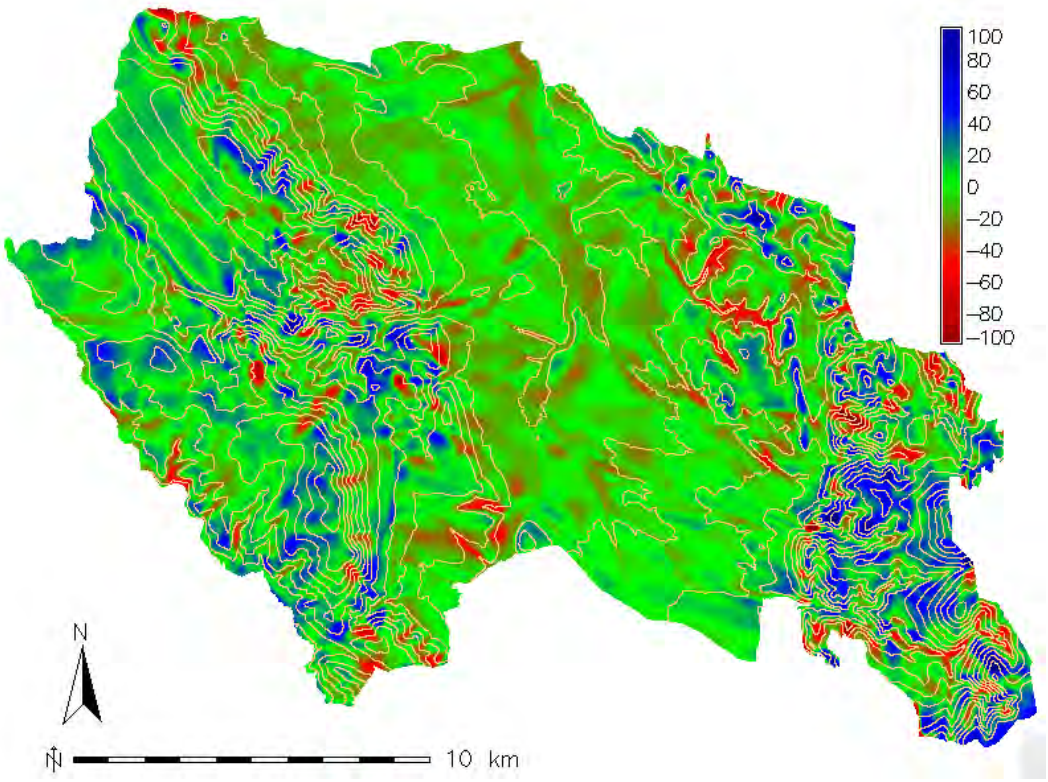
9.9. Podsumowanie

System GRASS jest elastycznym środowiskiem, w którym użytkownik może definiować własne rozszerzenia. Własne moduły mogą powstawać w języku C, C++. Mogą one też być tworzone jako skrypty albo w języku poleceń powłoki systemu operacyjnego albo w języku Python. GRASS udostępnia interfejsy programistyczne API dla języka Python pozwalające na dostęp do poleceń systemu GRASS, jak również na dostęp do danych zgromadzonych w bazie danych przestrzennych GRASS.

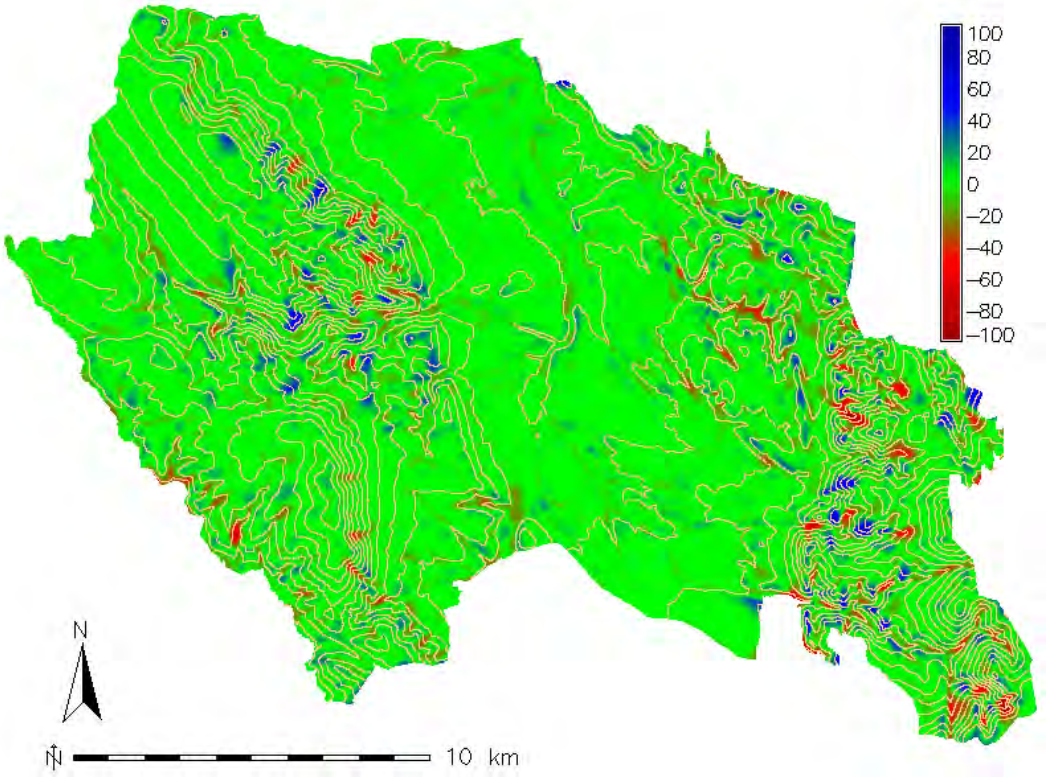
Przygotowane algorytmy interpolacji modelu wysokościowego na podstawie punktów losowo położonych wykorzystujące sztuczne sieci neuronowe wykazały podobną dokładność wyników interpolacji jak dokładność uzyskana za pomocą interpolacji funkcjami sklejalnymi.



Rys. 9.9 Rozkład przestrzenny błędów interpolacji funkcjami sklejalnymi (zbiór 3, algorytm BS_Bcub)



Rys. 9.10 Rozkład przestrzenny błędów interpolacji siecią neuronową (zbiór 3 , algorytm ANN)



Rys. 9.11 Rozkład przestrzenny błędów interpolacji metodą RST (zbiór 3 , algorytm RST)

Warto zauważyć, że interpolację wykonywano dla takich samych danych wejściowych, w rozumieniu zarówno zbiorów wybranych punktów, jak i wybranych cech (współrzędne X, Y). Prace poświęcone interpolacji wykorzystującej sztuczne sieci neuronowe wskazują na to, że lepsze rezultaty można osiągnąć używając innego przedstawienia danych wejściowych, wychodząc poza podawanie współrzędnych X,Y [2].

Przedstawione testy prowadzone były tylko w kierunku przedstawienia różnic w interpolacji pomiędzy algorytmami zawartymi w podstawowej wersji systemu GRASS a zaimplementowanymi algorytmami sztucznych sieci neuronowych.

Przygotowane moduły ANN dla systemu GRASS okazały się w pełni funkcjonalne i przydatne do interpolowania danych przestrzennych.

Literatura

[1] Gardner M.W., Droling S.R., 1998, *Artificial Neural Networks (the multilayer perceptron) - a review of application in the atmospheric sciences*, Atmospheric Environment, 32, 14/15, s. 2627-2636

[2] Hoseinali F., Delavar M.R., Amini J., Kianie M., 2005, *Interpolation of a Regular Grid from Irregular 3D Points, Using Artificial Neural Networks*. 8th Annual International Conference and Exhibition Map India 2005, New Delhi, ss. 9

[3] Karabork H., Baykan O. K., Altuntas C., Yildz F., 2008, *Estimation of unknown height with artificial neural network on digital terrain model*, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII, part B3b, s. 115-118

[4] Nissen S., 2003, *Implementation of a Fast Artificial Neural Network Library (FANN)*, Scientific Report, Department of Computer Science, University of Copenhagen, ss. 92

[5] Masters Timothy, 1996, *Sieci Neuronowe w praktyce*, Wydawnictwo Naukowo-Techniczne, ss. 197

[6] Osowski S., 1996, *Sieci neuronowe w ujęciu algorytmicznym*, Wydawnictwo Naukowo-Techniczne, ss. 178

[7] Rigol J.P., Jarvis C.H., Stuart N., 2001, *Artificial neural networks as a tool for spatial interpolation*, International Journal Geographical Information Science, vol 15, no. 4, s. 323-343

[8] Żurada J., Barski M., Jędruch W., 1996, *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, ss. 211

Źródła internetowe

[9] Strona główna projektu FANN, http://leenissen.dk/fann/

[10] Wykorzystanie języka Python w systemie GRASS, http://grass.osgeo.org/wiki/GRASS_and_Python

10. GRASS i R

Jacek Ślopek

10.1. System R

R jest systemem do obliczeń statystycznych i tworzenia wykresów naukowych [13]. Pomimo, iż jest on głównie narzędziem używanym w analizach danych, był i jest równie szeroko stosowany w modelowaniu i symulacjach. R jest również dobrze przygotowany do pracy z danymi zawartymi w systemach GIS. Poza klasycznymi metodami analiz statystycznych dostępnymi w podstawowej wersji systemu, użytkownicy mają również dostęp do uzupełniających jego możliwości bibliotek funkcji i pakietów użytecznych np. w analizie skupień, geostatystyce, analizach danych przestrzennych, czy choćby do przestrzennej ekonometrii [7, 8].

System R jest dostępny dla wszystkich użytkowników zgodnie z zasadami licencji GPL, co oznacza, że można go pobierać, instalować, użytkować i kopiować całkowicie za darmo we wszystkich zastosowaniach – a więc prywatnych, naukowych, czy biznesowych.

R jest jednocześnie środowiskiem pracy, jak również interpreterem specyficznego języka programowania. Powstał on pod wpływem dwóch istniejących już systemów: S (autorstwa Beckera, Chambersa i Wilksa z Laboratoriów Bella) [6] i Scheme (autorstwa Sussmana) [14]. Początkowe prace nad systemem prowadziło dwóch autorów – Ross Ihaka i Robert Gentleman na Wydziale Statystyki Uniwersytetu w Auckland w Nowej Zelandii.

R można zainstalować na komputerach wyposażonych w którykolwiek z najpopularniejszych systemów operacyjnych (Linux, MacOS, MS Windows). System domyślnie oferuje możliwość skorzystania z wielu procedur statystycznych, takich jak np. (uogólnione) modele liniowe, nieliniowe modele regresyjne, analiza serii czasowych, klasyczne parametryczne i nieparametryczne testy, czy analiza skupień. Dodatkowo, niektóre z modułów rozszerzających bazę funkcji systemowych zapewniają możliwość połączenia R z innymi programami czy systemami takimi, jak np. system GRASS.

Rdzeń systemu R obsługiwany jest przez interpretowany wewnętrzny język programowania zawierający instrukcje warunkowe i pętle, pozwalający tworzyć własne funkcje. Większość funkcji dostępnych dla użytkownika jest napisana w języku R. Niemniej istnieje możliwość użycia procedur z języków C, C++ lub FORTRAN w celu uzyskania większej wydajności w obliczeniach.

Charakterystyczne dla R jest to, że obiekty w systemie (zmienne, wyniki obliczeń, wczytane dane) są utrzymywane w trakcie sesji w tzw. przestrzeni roboczej (workspace). Przestrzeń ta, będąca obszarem pamięci, utrzymywana jest aż do końca sesji, gdy użytkownik może zdecydować, czy zapisać ją, by móc do niej powrócić podczas kolejnej sesji. Przestrzeń robocza będzie w tym przypadku zapisana na dysku do pliku o nazwie „.RData”, w katalogu, z którego był wywołany program R. Tym między innymi różni się R od swojego protoplasty – systemu S, gdzie obiekty już w trakcie sesji są zapisywane jako pliki na dysku twardym. Jedną z konsekwencji takiego podejścia jest większa szybkość pracy (dostęp do pamięci jest szybszy aniżeli dostęp do plików), choć jednocześnie inną konsekwencją jest to, że jeśli praca nie zostanie zapisana, a wydarzy się np. zawieszenie się oprogramowania, użytkownikowi grozi utrata rezultatów pracy. Przed wprowadzeniem wersji 1.2.0 systemu R, przestrzenie robocze były statyczne, obecnie

(aktualna wersja systemu to 2.12.0) ich wielkość ustalana jest dynamicznie [6]. Dodatkowo, poza zapisem przestrzeni roboczej, użytkownik ma również możliwość powrotu do wszystkich poleceń wydanych w trakcie sesji z R. Polecenia te zapisywane są w drugim ukrytym pliku o nazwie „.RHistory”.

10.2. Połączenie możliwości systemów GRASS i R

System GRASS stanowi zbiór ponad 350 modułów do obróbki i zarządzania danymi, przeprowadzania analiz i wizualizacji danych przestrzennych [4, 5, 9, 12]. Niemniej jego możliwości w niektórych dziedzinach nie są jeszcze kompletne. Takim obszarem jest np. geostatystyka z popularnymi metodami – krigingiem i cokrigingiem, których implementacja w systemie GRASS w aktualnej i stabilnej wersji oznaczonej numerem 6.4, jest jeszcze na etapie początkowym. Braki takie wymuszają wciąż korzystanie z dodatkowego oprogramowania. Na szczęście użytkownicy systemu GRASS mogą w pełni wykorzystać możliwości jakie daje im system R [10].

Możliwość połączenia R i GRASS pojawiła się w wersji 5.0 systemu GRASS. W pierwszej wersji interface pozwalał na pracę z danymi rastrowymi oraz punktowymi (zanim dane tego typu zostały zintegrowane z formatem wektorowym w GRASS 6). Nie było natomiast możliwości pracy z danymi wektorowymi. Połączenie R z GRASS zapewniał pakiet systemu R o nazwie GRASS. Po wprowadzeniu poważnych zmian w strukturze danych systemu GRASS, począwszy od wersji 6.0, za połączenie obu systemów odpowiada już inny pakiet: spgrass6 [4, 5]. Jego działanie opiera się na obsłudze klas z pakietu sp programu R związanych z danymi przestrzennymi. Instalacja pakietu spgrass6 wymaga także obecności w programie R zainstalowanych innych pakietów. Są to: wspomniany już pakiet sp, a także pakiety maptools i rgdal, gdyż GRASS używa bibliotek GDAL i OGR w swoim głównym mechanizmie importu/eksportu danych.

W celu wymiany danych pomiędzy R i GRASS stosowane są formaty pośrednie. Pakiet spgrass6 wykorzystuje formaty shapefile do danych wektorowych i BIL (Binary-Interleaved-by-Line) do danych rastrowych [2].

Użycie pakietu sp spowodowało także, że dostęp do danych przestrzennych systemu GRASS odbywa się prościej. Uproszczono również sposób wyświetlania tych danych w R – nie są już wymagane dodatkowe funkcje graficzne, które pozwalały na wyświetlanie danych przestrzennych systemu GRASS, gdy łączność zapewniał pakiet o nazwie GRASS [1, 2, 3].

Połączenie i współpraca obu systemów przebiega w kilku krokach:

- uruchomienie sesji w systemie GRASS
- uruchomienie z linii komend GRASS systemu R
- wczytanie w R pakietu odpowiedzialnego za komunikację pomiędzy GRASS i R
- odczyt i/lub zapis danych pochodzących z systemu GRASS, wykorzystanie poleceń R do przetworzenia danych przestrzennych
- zamknięcie systemu R i ewentualny zapis przestrzeni roboczej

Zapis przykładowej sesji w R wykorzystującej dane zapisane w systemie GRASS 6.4:

```
####################################
# Uruchomienie GRASS z linii komend
####################################

grass64

####################################
# Uruchomienie R
####################################

R

####################################
# Wczytanie biblioteki spgrass6
####################################

library(spgrass6)

####################################
# Wczytanie warstw z systemu GRASS
# i zapamietanie ich jako zmiennych
# w R
####################################
```

```
miejscowosci<-
   readVECT6("miejscowosci",
   ignore.stderr=TRUE)
uzytkowanie<-
   readRAST6("uz_bystrzyca",
   ignore.stderr=TRUE)
poziomice<-
   readVECT6("poziomice_50_bystrzyca"
   ,ignore.stderr=TRUE)
dtm<-readRAST6("dtm_bystrzyca_m",
   ignore.stderr=TRUE)

####################################
# Garsc informacji na temat
# wczytanych warstw
####################################

str(gmeta6())
summary(miejscowosci)
vInfo("miejscowosci")
vColumns("miejscowosci")
names(miejscowosci)
summary(uzytkowanie)
table(uzytkowanie$uz_bystrzyca)
summary(dtm)

####################################
# Wyrysowanie jednej z warstw
# na ekranie
####################################

image(uzytkowanie,"uz_bystrzyca",
   col=c("#c80000","#008000",
   "#0000c8","#808080","#c88000",
   "#00ff00"),axes=TRUE)
grid(col=colors()[200])
legend("bottomleft",
   legend=c("Obszary zabudowane",
   "Lasy","Rzeki","Nieużytki",
   "Grunty orne","Użytki zielone"),
   fill=c("#c80000","#008000",
   "#0000c8","#808080","#c88000",
   "#00ff00"),cex=0.8,bty="o",
   horiz=FALSE)
title("Zagospodarowanie terenu w
   gminie Bystrzyca Kłodzka")

####################################
# Porownanie z monitorem graficznym
# w GRASS
####################################

system("d.mon x0")
system("d.rast uz_bystrzyca")

####################################
# Wyrysowanie i zapisanie drugiej
# warstwy do pliku graficznego
####################################

png(filename=
   "orografia_bystrzyca.png",
   width=1000,height=1000,units="px",
   pointsize=10)
image(dtm,col=terrain.colors(21),
   axes=TRUE)
plot(poziomice,col="black",lwd=.5,
   add=TRUE)
plot(miejscowosci,pch=16,col="red3",
   axes=TRUE,add=TRUE)
grid(col=colors()[200])
legend("bottomleft",
   legend=seq(1300.0,300.0,by=-50),
   fill=rev(terrain.colors(21)),
   cex=1.5,bty="n",horiz=FALSE,
   bg=colors()[1])
text(3688000, 5458330, "m n.p.m.",
   cex=1.2,col="black")
title("Orografia gminy Bystrzyca
   Kłodzka")
dev.off()

####################################
# Zakonczenie pracy w R
####################################

q()
```

W trakcie przykładowej sesji w R wykorzystano kilka poleceń systemu R pozwalających na odczytanie i wyświetlenie danych przestrzennych pobranych z bazy systemu GRASS. Po uruchomieniu sesji GRASS, uruchomiony został system R (polecenie R). Następnie wczytano w nim bibliotekę spgrass6 odpowiedzialną za współpracę obu systemów (polecenie library). Poleceniem str można sprawdzić wewnętrzną strukturę obiektów odczytywanych w R. W sesji przykładowej obiekt ten zawierał metadane bieżącej lokacji systemu GRASS (odczytane za pomocą funkcji gmeta6), tj. położenie bazy danych GRASS, nazwę lokacji, mapsetu, rodzaj kodowania znaków w systemie, zasięgu regionu geograficznego, rozdzielczość mapy, ilość rastrów i informacje dotyczące odwzorowania w bieżącej lokacji.

Polecenia readVECT6 i readRAST6 pozwalają na wczytanie warstw rastrowych i wektorowych systemu GRASS. W przykładowej sesji dane zostały zapisane do zmiennych tymczasowych systemu R (noszą one nazwy „miejscowosci”, „dtm” i „uzytkowanie”).

Polecenia vInfo, vColumns, names i summary pozwalają na uzyskanie informacji na temat badanych zmiennych (np. ilości punktów, nagłówków kolumn z atrybutami), a także na wyliczenie i wyświetlenie prostych statystyk obejmujących zakres danych, które znajdują się w zmiennych systemu R. Dzięki poleceniu table można uzyskać informacje mogące stanowić podstawę do wykreślenia histogramu wartości. W przykładowej sesji uzyskano dzięki temu informacje na temat ilości rastrów przypisanych do poszczególnych klas użytkowania terenu.

Dane systemu GRASS w przykładowej sesji wyświetlane były na ekranie za pomocą poleceń plot i image. Dodatkowe elementy,

takie jak legenda, siatka współrzędnych, czy opisy tekstowe można umieścić na ekranie graficznym np. za pomocą poleceń title, grid, text czy legend.

W trakcie sesji przykładowej wykorzystano zarówno możliwość wyświetlania danych na ekranie graficznym w R, jak również zapisu wynikowego wykresu (mapy) do pliku graficznego na dysku (polecenia png/dev.off).

Po uruchomieniu systemu R w trakcie sesji GRASS polecenia systemu GRASS (ale również polecenia języka powłoki) mogą być wykonywane jedynie poprzez polecenie pośredniczące: system. W przykładowej sesji skorzystano z tego polecenia, by otworzyć dodatkowo monitor graficzny systemu GRASS i wyświetlić w nim wskazaną warstwę rastrową. Zakończenie pracy w systemie R następuje po wydaniu polecenia q().

10.3. Użycie R w trybie wsadowym

Podobnie jak w systemie GRASS, prawdopodobnie w większości przypadków użytkownicy R korzystają interaktywnie z tego oprogramowania. Niemniej, zarówno w przypadku GRASS, jak i w R ukryte są dodatkowe możliwości, jakie daje nam wsadowy tryb pracy. R, tak jak GRASS można uruchomić za pomocą skryptu i zautomatyzować obliczenia. Tryb wsadowy i uruchamianie obliczeń za pomocą skryptów może być użyteczne w wielu sytuacjach. Dzięki skryptom można uruchomić program do długotrwałych obliczeń, zaplanować uruchomienie w najbardziej sprzyjającym nam czasie (np. w nocy, gdy komputer lub serwer nie jest mocno obciążony), wykorzystując przy okazji możliwości, jakie daje terminarz zadań w systemie operacyjnym. Tryb wsadowy jest przydatny również wtedy, gdy np. chcemy wykorzystać system R zainstalowany na serwerze do wykonania obliczeń zlecanych poprzez użytkowników korzystających ze stron www, na których wyniki obliczeń są prezentowane. Na samej prezentacji wyników się nie kończy – skrypty można bowiem tak skonstruować, by użytkownik mógł na stronie www wyposażonej w formularz wprowadzać swoje zmienne, które następnie będą wykorzystywane w obliczeniach.

Tryb wsadowy jest dostępny zarówno dla użytkowników systemów Unix/Linux, jak i systemu MS Windows. W obu systemach wykorzystuje się potokowanie oraz specjalny tryb pracy R wywoływany w linii poleceń opcjami „vanilla” i „no-save”, dodawanymi po poleceniu „R”. Druga z wymienionych opcji powoduje, że na dysku nie zapisuje się dodatkowo plik „.RData” z przestrzenią roboczą, zawierającą historię poleceń oraz wszystkie dane wykorzystywane w obliczeniach. Ma to istotne znaczenie, jeśli w miejscu pracy mamy limitowaną przestrzeń dyskową do dyspozycji.

```
R --no-save --vanilla < skrypt.r >
   wyniki_obliczen.txt
```

Komenda pozwalająca uruchomić R w trybie wsadowym w systemie Linux (skrypt.r jest plikiem z poleceniami R, a plik wyniki_obliczen.txt jest plikiem, w którym zapisane zostaną wyniki obliczeń)

By móc wykorzystać tryb wsadowy w środowisku MS Windows i zapisać wyniki w wybranym przez użytkownika katalogu, należy w nim stworzyć i uruchomić skrypt *.bat, który uruchomi opcje „vanilla” i „no-save” terminala R. Jest to możliwe poprzez użycie dodatkowego programu Rterm.exe, który zawarty jest w systemie R przygotowanym dla MS Windows. Po uruchomieniu Rterm.exe z opcjami „vanilla” i „no-save” możliwe będzie uruchomienie skryptu przygotowanego dla systemu R i zapisanie wyników pracy w pożądanym katalogu.

```
C:\program files\R\rw1080\bin\Rterm
   --vanilla –no-save
```

Zawartość skryptu R_vanilla.bat, włączającego tryb „vanilla” z opcją „no-save” (Program Rterm.exe znajduje się w tym przykładzie w katalogu c:\Program Files\R\rw1080\bin)

```
R < skrypt.r > wyniki_obliczen.txt
```

Komenda pozwalająca uruchomić R w trybie wsadowym w systemie MS Windows (skrypt.r jest plikiem z poleceniami R, a plik

wyniki_obliczen.txt jest plikiem, w którym zapisane zostaną wyniki obliczeń). Komendę tę należy wydać będąc w katalogu roboczym.

Tryb wsadowy pozwala również na generowanie plików graficznych. Jeśli w trakcie pracy użytkownik tworzy wykresy, może skorzystać z możliwości jakie daje system R i zapisać wyniki w postaci plików graficznych na dysku. W R można zapisać wykresy zarówno w formatach rastrowych (bitmapy), jak również w formatach wektorowych (np. postscript, pdf).

Aby móc skorzystać z trybu wsadowego systemu R i skryptów wykorzystujących połączenie z systemem GRASS należy uruchomić tryb wsadowy po uruchomieniu systemu GRASS (w trakcie jego sesji).

Zawartość przykładowego skryptu dla trybu wsadowego:

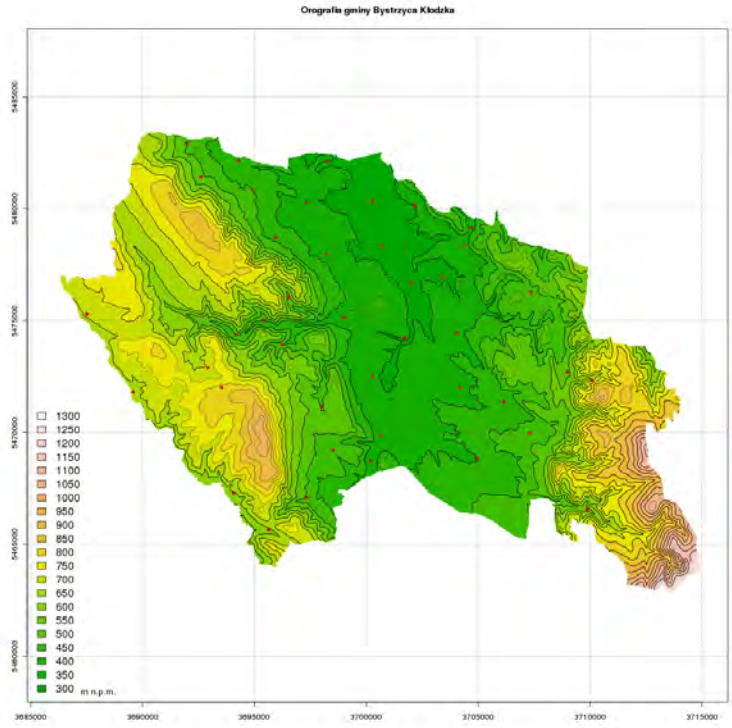
```
####################################
# Generowanie plikow png z mapami
# orografii gminy oraz uzytkowania
# terenu
# Skrypt dla trybu wsadowego:
#   bystrzyca_dtm_uz.r
####################################

library(spgrass6)

miejscowosci<-
   readVECT6("miejscowosci",
   ignore.stderr=TRUE)
poziomice<-
   readVECT6("poziomice_50_bystrzyca"
   , ignore.stderr=TRUE)
uzytkowanie<-
   readRAST6("uz_bystrzyca",
   ignore.stderr=TRUE)
dtm<-readRAST6("dtm_bystrzyca_m",
   ignore.stderr=TRUE)
str(gmeta6())
summary(miejscowosci)
vInfo("miejscowosci")
vColumns("miejscowosci")
names(miejscowosci)
summary(uzytkowanie)
table(uzytkowanie$uz_bystrzyca)
summary(dtm)

png(filename="orografia.png",
   width=1000,height=1000,
   units="px",pointsize=10)
image(dtm,col=terrain.colors(21),
   axes=TRUE)
plot(poziomice,col="black",lwd=.5,
   add=TRUE)
plot(miejscowosci,pch=16,col="red3",
   axes=TRUE,add=TRUE)
grid(col=colors()[200])
legend("bottomleft",
   legend=seq(1300.0,300.0,by=-50),
   fill=rev(terrain.colors(21)),
   cex=1.5,bty="n",horiz=FALSE,
   bg=colors()[1])
text(3688000, 5458330, "m n.p.m.",
   cex=1.2,col="black")
title("Orografia gminy Bystrzyca
   Kłodzka")
dev.off()

png(filename="uzytkowanie.png",
   width=1000,height=1000,units="px",
   pointsize=10)
image(uzytkowanie,"uz_bystrzyca",
   col=c("#c80000","#008000",
   "#0000c8","#808080","#c88000",
   "#00ff00"),axes=TRUE)
grid(col=colors()[200])
legend("bottomleft",
   legend=c("Obszary zabudowane",
   "Lasy","Rzeki","Nieużytki",
   "Grunty orne","Użytki zielone"),
   fill=c("#c80000","#008000",
   "#0000c8","#808080","#c88000",
   "#00ff00"),cex=2.0,bty="o",
   horiz=FALSE)
title("Zagospodarowanie terenu w
   gminie Bystrzyca Kłodzka")
dev.off()

rm(list = ls(all = TRUE))
```

Ostatnia linia w skrypcie nie jest konieczna, jeśli korzystamy z opcji „no-save” przy wywołaniu R. Jeśli jednak jej nie użyliśmy, a chcemy wyczyścić pamięć i zapobiec zapisaniu się pliku „.RData” na dysku, należy tę linię dodać w swoim skrypcie [4]. Na rysunkach 10.1 i 10.2 przedstawiono dwie przykładowe mapy uzyskane w systemie R po wywołaniu omawianego skryptu, wykorzystującego bibliotekę spgrass6.

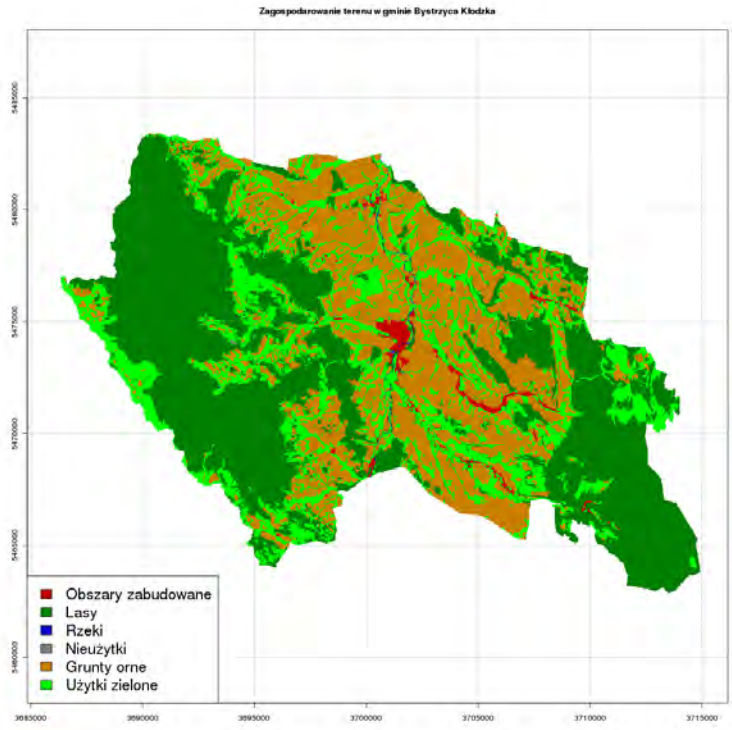
Tryb wsadowy poprzez mechanizm potokowania, wysyła wszystkie wyniki obliczeń (wywołań poleceń systemu) do pliku tekstowego. Wewnątrz pliku zapisane są wszystkie informacje, jakie standardowo użytkownik zobaczyłby na ekranie swojego komputera, gdyby pracował z systemem R interaktywnie. Będą to zatem nie tylko wyniki działania poleceń, ale również i ich treść oraz, jeśli w skrypcie będą błędy, komunikaty systemowe.

10.4. Wykorzystanie w systemie GRASS ilustracji utworzonych w systemie R

Wspomniana wcześniej możliwość zapisu wykresów jako plików graficznych pozwala użytkownikom systemu GRASS na inne połączenie obu systemów.



Rys. 10.1 Mapa orografii gminy Bystrzyca Kłodzka uzyskana w systemie R za pomocą skryptu wykorzystującego połączenie z systemem GRASS. Czerwonymi punktami oznaczono lokalizację miejscowości na terenie gminy.



Rys. 10.2 Mapa użytkowania terenu w gminie Bystrzyca Kłodzka uzyskana w systemie R za pomocą skryptu wykorzystującego połączenie z systemem GRASS.

Przykładowo, na stronach Wiki GRASS prezentowane są mapy tematyczne obejmujące obszar Belgii, przy których powstawaniu wykorzystano możliwości generowania wykresów i zapisu plików wektorowych (PostScript) w systemie R. Pliki takie mogą być z powodzeniem wykorzystane w systemie GRASS jako element ilustracyjny na mapach tworzonych przy użyciu polecenia ps.map lub ps.output. Przykłady map tworzonych w systemie GRASS a ilustrowanych wykresami pochodzącymi z R można znaleźć w sieci Internet [11].

10.5. Podsumowanie

Na przydatność użycia GRASS w parze z systemem R wskazywali już użytkownicy systemu GRASS w wersji 5. Wersja ta pojawiła się w powszechnym użyciu około roku 2000. W roku 2004 Wegmann i Lennert [15] przeprowadzili rozpoznanie wśród użytkowników systemu GRASS, z którego wynikało, iż R zajmuje drugie w kolejności miejsce pośród dodatkowego oprogramowania używanego razem z system GRASS. Od tego czasu interfejs umożliwiający pracę z danymi przestrzennymi systemu GRASS w R ulegał stałym modyfikacjom i poprawkom, co zaowocowało m. in. możliwością pracy z danymi wektorowymi w formacie obecnym w GRASS 6.x. Pojawienie się nowszych, rozszerzonych wersji GRASS i stały rozwój jego możliwości obliczeniowych nie zmniejszył roli pakietu R i stanowi on wciąż cenne uzupełnienie oraz popularne narzędzie wykorzystywane w analizach przestrzennych wśród użytkowników GRASS. Nie bez znaczenia pozostaje również fakt, że można stosować R w trybie wsadowym, ułatwiającym tworzenie skryptów dla systemu GRASS. Skrypty te pozwalają na przeprowadzanie zautomatyzowanych obliczeń na dużych pakietach danych, a także na zastosowanie połączenia możliwości obu systemów w obróbce i prezentacji danych na stronach www.

Literatura

[1] Bivand R. S., 2005, *Interfacing GRASS 6 and R*, GRASS-News, Vol. 3, June 2005, s. 11-15

[2] Bivand R. S., 2007, *Using the R-GRASS interface*, OSGeo Journal, Vol. 1, May 2007, s. 36-38

[3] Bivand R. S., Pebesma E. J., Gómez-Rubio V., 2008, *Applied Spatial Data Analysis with R*, Springer Science+Business Media, LLC, New York, ss. 374

[4] Neteler M., Mitasova H., 2004, *Open Source GIS – A GRASS GIS approach (2nd ed)*, Kluwer Academic Publishers, Bostonss. ss. 424

[5] Neteler M., Mitasova H., 2008, *Open Source GIS – A GRASS GIS approach (3rd ed)*, Springer Science+Business Media, LLC, New York, ss. 406

[6] Venables W. N. and Ripley B. D., 2002, *Modern Applied Statistics with S (4th ed)*, Springer, New York, ss. 495

Źródła internetowe

[7] Bivand, R. S., Gebhardt, A., 1998, *Implementing functions for spatial statistical analysis using the R language*. 38 Congress of the European Regional Science Association, Vienna, 28 August - 1 September 1998, http://www.math.uni-klu.ac.at/stat/users/agebhard/ERSA-98-D8-427/rpaper1.html

[8] Bivand, R. S., Neteler, M., 2000, *Open Source geocomputation: using the R data analysis language integrated with GRASS GIS and PostgreSQL data base systems*. Proceedings 5th conference on GeoComputation, 23-25 August 2000, University of Greenwich, U.K., http://reclus.nhh.no/gc00/geocomp2000.pdf

[9] Dassau O., Holl S., Neteler M., Redslob M., 2005, *An introduction to the practical use of the Free Geographical Information System GRASS 6.0 (Version 1.2)*, GDF Hannover bR, http://www.gdf-hannover.de/lit_html/grass60_v1.2_en/grass60_engl.html
[10] Neteler M., 2000-2007, *Short Introduction to Geostatistical and Spatial Data Analysis with GRASS and R statistical data language*, http://grass.fbk.eu/statsgrass/grass_geostats.html
[11] Przykładowe wykresy generowane w systemie R, jako ilustracja na mapach wykonywanych dzięki poleceniu ps.map w GRASS, http://geog-pc40.ulb.ac.be/grass/psmap/
[12] Strona projektu GRASS, http://grass.osgeo.org/
[13] Strona projektu R, http://www.r-project.org
[14] Sussman G.J., Steel G.L., 1975, SCHEME an interpreter for extended lambda calculus, http://repository.readscheme.org/ftp/papers/ai-lab-pubs/AIM-349.pdf
[15] Wegmann M., Lennert M. (2005), GRASS User Survey 2004, GRASS-News Vol.2, s. 2-16, http://grass.fbk.eu/newsletter/GRASSNews_vol2.pdf