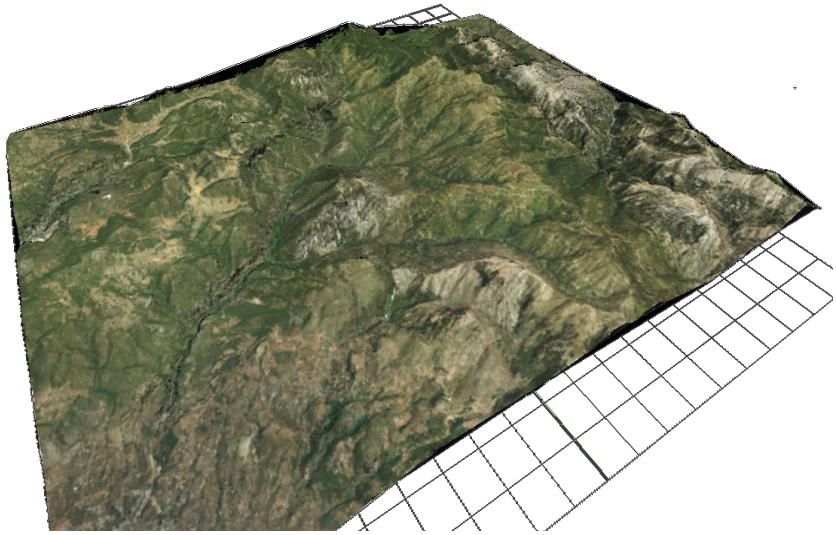
National Technical University of Athens

INTERMEDIATE – INTERVENTIONAL
POSTGRADUATE PROGRAMME
“WATER RESOURCES SCIENCE AND TECHNOLOGY”

DEVELOPMENT OF A DISTRIBUTED HYDROLOGICAL SOFTWARE APPLICATION EMPLOYING NOVEL VELOCITY - BASED TECHNIQUES



Konstantina Risva

Athens, November 2018

“WATER RESOURCES SCIENCE AND TECHNOLOGY”

Supervisor: A. Efstratiadis

National Technical University of Athens

INTERMEDIATE – INTERVENTIONAL POSTGRADUATE PROGRAMME
"WATER RESOURCES SCIENCE AND TECHNOLOGY"

DEVELOPMENT OF A DISTRIBUTED HYDROLOGICAL SOFTWARE APPLICATION EMPLOYING NOVEL VELOCITY - BASED TECHNIQUES

Konstantina Risva

Athens, November 2018

Supervisor: A. Efstratiadis

Ευχαριστίες

Γράφοντας αυτές τις τελευταίες γραμμές της εργασίας αυτής νιώθω την ανάγκη αρχικά να ευχαριστήσω το Δρ. Πολιτικό Μηχανικό Ανδρέα Ευστρατιάδη, μέλος ΕΔΙΠ της σχολής Πολιτικών Μηχανικών Ε.Μ.Π. για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου ένα ιδιαίτερα ενδιαφέρον και συνάμα απαιτητικό θέμα. Χωρίς την πολύτιμη βοήθεια του και τη συνεχή του καθοδήγηση η εργασία αυτή δεν θα ήταν ίδια. Νιώθω ιδιαίτερα ευγνώμων στο φίλο πλέον Ανδρέα που μου χάρισε απλόχερα όλη τη γνώση του στο γνωστικό αντικείμενο της υδρολογίας και των πλημμυρών στα μικρά εντατικά μαθήματα που μου πρόσφερε στο γραφείο 208. Χάρη στην ανεξάντλητη υπομονή του και τη διαρκή του παρακίνηση αντιμετώπιζα τις μικρές απογοητεύσεις και δυσκολίες στην πορεία μου προς την ολοκλήρωση της εργασίας.

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στον κ. Α. Κουκοβίνο Αγρονόμο και Τοπογράφο Μηχανικό, για τη στήριξη και τις καίριες συμβουλές του, που αποδείχτηκαν ιδιαίτερα χρήσιμες στη πορεία της εργασίας, καθώς και στις Ε. Σαββίδου, Ε. Μιχαηλίδη και της Σ. Αντωνιάδη που συνέβαλαν σε μεγάλο βαθμό με τις επιστημονικές τους εργασίες και τη βοήθειά τους στην επιτυχή ολοκλήρωσης αυτής της εργασίας.

Ευχαριστώ ιδιαίτερα τον Αναπληρωτή Καθηγητή Ν. Μαμάση της Σχολής Πολιτικών Μηχανικών Ε.Μ.Π. και την Επίκουρη Καθηγήτρια κ. Αικατερίνη Νάνου, μέλη της τριμελούς επιτροπής για την τις πολύτιμες συμβουλές τους.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την εμπιστοσύνη και τη διαρκή τους καθοδήγηση για την ολοκλήρωση των μεταπτυχιακών μου σπουδών. Ιδιαίτερα ευχαριστώ τον αδελφό μου, που αν και 2000 χιλιόμετρα μακριά , ήταν πάντα το στήριγμά μου στις δύσκολες στιγμές. Ένα ευχαριστώ οφείλω στους φίλους μου για τη δύναμη που μου έδιναν κάθε φορά που απογοητευόμουν. Τέλος, δεν θα μπορούσα να παραλείψω να ευχαριστήσω τον υποψήφιο διδάκτορα Διονύση Νικολόπουλο χωρίς τον οποίο η ολοκλήρωση της εργασίας αυτής δεν θα ήταν εφικτή. Η απεριόριστη βοήθεια και υποστήριξη που μου έδινε σε κάθε βήμα είναι αδύνατο να περιγράφουν σε λίγες γραμμές.

Κωνσταντίνα Ρίσβα,
Νοέμβριος 2018

Abstract

The aim of this study is the development of an event-based distributed hydrological model, incorporating novel methodologies for estimating the effective rainfall and representing the routing processes. First, we distinguish the effective from the gross rainfall, at a cell basis, thus extracting the spatial distribution of surface runoff during the simulation period. The underlying model is based on an improved NRCS-CN scheme, which uses a spatially-varying CN (different for each cell) and two lumped dimensionless parameters, i.e. one for representing the antecedent soil moisture conditions (AMC) of the basin at the beginning of the storm event, and one for estimating the initial rainfall abstraction. Key modelling novelty is the adjustment of the so-called reference CN value (i.e. the value that refers to average soil moisture conditions and 20% abstraction ratio) against the two aforementioned lumped parameters. For the propagation of runoff to the basin outlet two flow types are considered, i.e. an overland flow across the catchment's terrain, and a channel flow along the river network. These are synthesized by employing a velocity-based approach, to determine the flood hydrograph. This approach implements an original methodology for assigning realistic velocity values along the river network. These use macroscopic hydraulic information as well as the time of concentration of the basin, which is considered function of runoff intensity. The proposed approach takes advantage of regional relationships and literature values for assigning appropriate values to all model attributes, except for the two lumped parameters of the rainfall-runoff transformation, which are either manually assigned or inferred through calibration. In the last case, it is essential to extract the sub-surface flow component (interflow) from the total hydrograph, which may be done through several approaches of varying complexity. Here we propose an empirical method, requiring the fitting of a lumped hydrological model the observed hydrograph, which explicitly accounts for the contribution of interflow to total runoff. An alternative, more integrated approach, aims at running the distributed model with additional functionalities, in order to obtain the full hydrograph at the basin outlet. In this context, we have also developed a more generic version of the modeling framework, in which the NRCS-CN procedure is combined with a continuous soil moisture accounting scheme, thus generating both the surface (overland) runoff as well as the interflow through the unsaturated zone. Apparently, this augmented version requires few additional parameters, since more processes are accounted for within the simulation procedure. For the schematization of the model domain, the user needs to formulate two spatial layers, i.e. a grid-based partition of the basin to equally-dimensioned (squared) cells, and a graph- based configuration of the hydrographic network, comprising junctions and interconnected river segments. In the context of model development, we used the high-level programming language, Python, to build a GUI interface, for data management and visualization, and to run simulations and optimizations. The two modeling versions and the software are successfully tested in the representation of two flood events across Nedontas river basin.

Keywords: rainfall-runoff modelling, Python & hydrology, improved NRCS-CN method, runoff intensity-dependent channel velocities, distributed event-based model

Εκτενής περίληψη

Ανάπτυξη κατανεμημένου υδρολογικού λογισμικού με εφαρμογή καινοτόμων κινηματικών τεχνικών

Εισαγωγή

Τα μοντέλα βροχής – απορροής, προσπαθούν να προσομοιώσουν μία από τις πιο περίπλοκες φυσικές διεργασίες, αυτή της μετατροπής της επιφανειακής βροχής σε υδρογράφημα απορροής. Όπως όλα τα μοντέλα, αποτελούν μία απλοποιημένη αναπαράσταση της πραγματικότητας. Στη βιβλιογραφία συναντώνται ποικίλα μοντέλα, τα οποία μπορούν συνοπτικά να κατηγοριοποιηθούν ανάμεσα σε:

- Αδιαμέριστα και κατανεμημένα: Τα αδιαμέριστα αντιμετωπίζουν όλη τη λεκάνη σαν μία ενότητα με κοινές φορτίσεις και κοινές παραμέτρους για όλη την έκταση ενώ τα κατανεμημένα επιχειρούν τη κατάτμηση σε πολλές μικρότερες χωρικές ενότητες με διαφορετική συμπεριφορά.
- Event-based (Ενός επεισοδίου) και συνεχόμενα μοντέλα: Τα πρώτα αφορούν μία συγκεκριμένη χρονική περίοδο, προκειμένου να παρουσιαστεί η απόκριση της λεκάνης απορροής στο συγκεκριμένο γεγονός βροχής. Αντιθέτως, τα συνεχόμενα μοντέλα προσομοιώνουν μεγάλες χρονικές περιόδους, που περιλαμβάνουν γεγονότα βροχόπτωσης και μη, καθώς και μεταβαλλόμενες συνθήκες κατά τη διάρκεια της προσομοίωσης. Συνήθως, τα event-based μοντέλα παρουσιάζουν ευαισθησία ως προς τις αρχικές συνθήκες, γεγονός που απαιτεί ιδιαίτερη προσοχή από το χρήστη κατά τον καθορισμό τους.
- Μοντέλα φυσικής βάσης (άσπρου κουτιού - white-box), εμπειρικά (μαύρου κουτιού - black-box) and εννοιολογικά μοντέλα. Τα πρώτα βασίζονται στη χωρική κατανομή των παραμέτρων και περιγράφουν τα φυσικά χαρακτηριστικά. Ωστόσο, συνήθως πρόκειται για περίπλοκα μοντέλα. Όσον αφορά τα εμπειρικά μοντέλα, αυτά μπορούν να προβλέπουν με ακρίβεια, όμως δεν μπορούν να εφαρμοστούν σε διαφορετική λεκάνη. Τέλος τα εννοιολογικά μοντέλα είναι παραμετρικά, βασίζονται σε ημι – εμπειρικές σχέσεις.
- Ντετερμινιστικά και στοχαστικά μοντέλα: Τα ντετερμινιστικά μοντέλα δημιουργούν το ίδιο αποτέλεσμα σε κάθε τρέξιμο του μοντέλου για δεδομένες τιμές παραμέτρων. Αντιθέτως, τα στοχαστικά, αξιοποιώντας πιθανοτικές κατανομές, παρέχουν διαφορετικά αποτελέσματα ανάλογα με τις παραμέτρους που έχουν χρησιμοποιηθεί.

Μεθοδολογική προσέγγιση

Στην παρούσα εργασία αναπτύσσεται ένα μοντέλο βροχής – απορροής προσομοίωσης ενός μεμονωμένου επεισοδίου (event-based), χρησιμοποιώντας κυρίως μία κατανεμημένη (distributed) προσέγγιση αξιοποιώντας παράλληλα και αδιαμέριστες παραμέτρους για κάποιες διαδικασίες, καθώς και διάφορες τεχνικές για την αναπαράσταση των διεργασιών, τόσο εννοιολογικές όσο και φυσικής βάσης.

Ως πρωταρχικό στάδιο, γίνεται διαχωρισμός της ενεργού βροχόπτωσης, σε επίπεδου κελιού, εξάγοντας κατά αυτόν τον τρόπο τη χωρική κατανομή της επιφανειακής απορροής στην εξεταζόμενη περίοδο προσομοίωσης. Το προτεινόμενο μοντέλο βασίζεται σε μία βελτιωμένη εκδοχή της NRCS-CN μεθόδου, χρησιμοποιώντας μία τιμή αναφοράς CN η οποία είναι διαφορετική για κάθε κελί και δύο αδιάστατες παραμέτρους, κοινές για ολόκληρη τη λεκάνη.

Πιο συγκεκριμένα, η μία αντιπροσωπεύει τις συνθήκες υγρασίας του εδάφους κατά την έναρξη του γεγονότος και η δεύτερη την αρχική κατακράτηση του εδάφους. Η προτεινόμενη μεθοδολογία έχει πολλές καινοτομίες σχετικά με την εκτίμηση της τιμή αναφοράς CN από γεωχωρικά δεδομένα και την αναπροσαρμογή της μέγιστης δυνητικής κατακράτησης σε σχέση με τις δύο παραμέτρους του μοντέλου.

Προκειμένου να εκτιμηθεί η απορροή στην έξοδο της λεκάνης απορροής διαχωρίζονται δύο βασικοί τύποι ροής: η επιφανειακή στην επιφάνεια της λεκάνης και η ροή στο υδρογραφικό δίκτυο. Αξιοποιώντας μία νέα προσέγγιση βασιζόμενη στην εκτίμηση των ταχυτήτων σε κάθε κελί (velocity-based approach) και συνδυάζοντας τους δύο τύπους ροής συνθέτεται το πλημμυρογράφημα στην έξοδο της λεκάνης. Στην παρούσα προσέγγιση στα κελιά που ανήκουν στο υδρολογικό δίκτυο της περιοχής γίνεται μία ψευδο-υδραυλική εκτίμηση η οποία σε αντίθεση με τις έως τώρα προσεγγίσεις, διορθώνει την εκτιμώμενη ταχύτητα που προκύπτει από το χρόνο συγκέντρωσης της λεκάνης σε κάθε κλάδο του δικτύου, ανάλογα τα φυσικά χαρακτηριστικά του κλάδου (κλίση, τραχύτητα). Με αυτό τον τρόπο δίνονται ρεαλιστικές τιμές ταχύτητας στα τμήματα του υδρογραφικού δικτύου που διαφέρουν χωρικά. Ακόμη, γίνεται διόρθωση της ταχύτητας λόγω χρονικής μεταβλητότητας με βάση την ένταση απορροής του επεισοδίου, αφού λαμβάνεται επιπρόσθετα υπόψη η σύγχρονη βιβλιογραφία που υποδεικνύει ότι ο χρόνος συγκέντρωσης μίας λεκάνης απορροής δεν είναι ένα σταθερό μέγεθος, αλλά μεταβαλλόμενο ανάλογα με την ένταση της απορροής και δίνονται προσεγγιστικές σχέσεις.

Η προτεινόμενη μεθοδολογία, αξιοποιώντας τις χωρικές σχέσεις καθώς και τις βιβλιογραφικές τιμές θέτει κατάλληλες τιμές σε όλα τα χαρακτηριστικά μεγέθη του μοντέλου, εκτός από τις δύο κοινές παραμέτρους που αφορούν όλη τη λεκάνη. Οι τελευταίες, είτε ορίζονται με εκτίμηση του μελετητή, είτε προκύπτουν μέσω βαθμονόμησης δεδομένου ότι υπάρχουν διαθέσιμα δεδομένα παρατηρημένης απορροής. Στην περίπτωση που οι παράμετροι εκτιμούνται μέσω βαθμονόμησης, είναι απαραίτητος ο διαχωρισμός της υποδερμικής ροής από το συνολικό υδρογράφημα. Στη βιβλιογραφία συναντώνται ποικίλες διαδικασίες διαχωρισμού διαφορετικής πολυπλοκότητας. Στο πλαίσιο της παρούσας εργασίας προτείνεται μία εμπειρική μέθοδος, που απαιτεί την προσαρμογή ενός αδιαμέριστου υδρολογικού μοντέλου στο παρατηρημένο υδρογράφημα, ώστε να εκτιμηθεί η συμβολή της βασικής ροής στο συνολικό υδρογράφημα και να αφαιρεθεί.

Επιπρόσθετα όμως, δημιουργείται και ένα δεύτερο κατανεμημένο μοντέλο, το οποίο προσομοιώνει και την υποδερμική ροή μέσω ενός μοντέλου γραμμικού ταμιευτήρα σε κάθε κελί για την εξαγωγή του ολικού υδρογραφήματος στην έξοδο της λεκάνης. Αναπτύχθηκε κατά αυτόν τον τρόπο μία πιο γενικευμένη εκδοχή του μοντέλου, στην οποία η NRCS-CN διαδικασία συνδυάζεται με την προσομοίωση της μεταβολής της υγρασία του εδάφους στο χρόνο, δημιουργώντας την επιφανειακή αλλά και την υποδερμική απορροή. Όπως είναι αναμενόμενο αυτό το πιο ολοκληρωμένο μοντέλο (συνολικό μοντέλο) έχει περισσότερες απαιτήσεις παραμέτρων, λόγω των περισσότερων διεργασιών.

Προκειμένου να σχηματοποιηθεί το μοντέλο, απαιτούνται αρχικά η λεκάνη απορροής και το υδρογραφικό δίκτυο σε επίπεδο κελιού. Τα δύο αυτά επίπεδα μπορούν να εξαχθούν με βάση το Ψηφιακό Μοντέλο Εδάφους (Digital Elevation Model – DEM) της λεκάνης, χρησιμοποιώντας τα διαθέσιμα εργαλεία σε περιβάλλον Γεωγραφικών Συστημάτων Πληροφοριών (Geographical Information System – GIS). Για την εκτίμηση των χωρικών παραμέτρων του μοντέλου, είναι απαραίτητα τα εξής γεωγραφικά δεδομένα:

- Γεωλογικοί χάρτες
- Χάρτες υδροπερατότητας
- Χάρτες χρήσεων/κάλυψης γης
- Χάρτες κλίσεων

- Τμήματα του υδρογραφικού δικτύου και γεωμετρικά χαρακτηριστικά τους

Ακόμη, ο χάρτης χωρικής κατανομής της βροχόπτωσης για μία δεδομένη χρονική περίοδο είναι απαραίτητο στοιχείο εισόδου του μοντέλου. Στην περίπτωση της βαθμονόμησης, είτε το πλημμυρογράφημα (για το επιφανειακό μοντέλο) είτε το συνολικό υδρογράφημα (για το συνολικό μοντέλο) θεωρούνται αναγκαία.

Κατάστρωση εξισώσεων μοντέλων

Αξιοποιώντας την παρακάτω εμπειρική σχέση υπολογίζεται το ενεργό ύψος βροχής. Τα μεγέθη που αναφέρονται είναι αθροιστικά.

$$h_e = \begin{cases} 0 & h \le h_{a0} \\ \dfrac{(h - h_{a0})^2}{h - h_{a0} + S} & h > h_{a0} \end{cases} \quad \text{Εξ.1}$$

$$h_{a0} = \lambda S \quad \text{Εξ. 2}$$

Όπου S: η μέγιστη δυνητική κατακράτηση και λ αδιάστατη παράμετρος που εκφράζει ποσοστό.

Σύμφωνα με την κλασική προσέγγιση κατά την SCS, η μέγιστη δυνητική κατακράτηση δίνεται από τη σχέση:

$$S = 254(\frac{100}{CN} - 1) \quad \text{Εξ. 3}$$

Κατά τα πρότυπα της SCS η ποσότητα CN εξαρτάται από τα χαρακτηριστικά του εδάφους καθώς και από την εδαφική υγρασία κατά την έναρξη του γεγονότος. Διαχωρίζονται τρεις καταστάσεις εδαφικής υγρασίας (Τύπου I: Ξηρή, Τύπου II: Μέση, Τύπου III: Υγρή). Η Εξίσωση 3 αναφέρεται σε μέση εδαφική υγρασία. Για λόγους ευκολίας οι τιμές του CN που συναντώνται στη βιβλιογραφία αναφέρονται σε μέσες συνθήκες υγρασίας. Για τους άλλους δύο τύπους υγρασίας χρησιμοποιούνται οι παρακάτω εξισώσεις:

$$CN_I = \frac{4.2CN_{II}}{10 - 0.058CN_{II}} \quad \text{Εξ. 4}$$

$$CN_{III} = \frac{23CN_{II}}{10 + 0.13CN_{II}} \quad \text{Εξ. 5}$$

Ωστόσο, στο πλαίσιο της παρούσας εργασίας το CN για τις μέσες συνθήκες υγρασίας υπολογίζεται για κάθε κελί σε περιβάλλον GIS από την παρακάτω σχέση:

$$CN_{II} = 10 + 9 \times i_{PERM} + 6 \times i_{VEG} + 3 \times i_{SLOPE} \quad \text{Εξ. 6}$$

Όπου i_{PERM}, i_{VEG} και i_{SLOPE} οι τρεις κλάσεις εδάφους, περατότητα, χρήσεις γης και δυνατότητα αποστράγγισης, που προκύπτουν από πίνακες και κυμαίνονται από 1 έως 5.

Κρίνεται αναγκαίο οι τιμές που προκύπτουν για το CN να διορθώνονται ώστε να λαμβάνεται υπόψη η κατάσταση υγρασίας και πριν το γεγονός βροχής. Κατά αυτό τον τρόπο για οποιεσδήποτε συνθήκες υγρασίας που εκφράζονται με ποσοστό (0 – 1) αξιοποιείται ο παρακάτω τύπος:

$$CN_{cor} = \begin{cases} CN_{II} - \dfrac{CN_{II} - CN_I}{0.4}\left(0.5 - AMC_{coef}\right), AMC_{coef} < 0.5 \\ CN_{III} + \dfrac{CN_{III} - CN_{II}}{0.4}\left(AMC_{coef} - 0.5\right), AMC_{coef} \geq 0.5 \end{cases}$$ Εξ. 7

Από τις εκτιμημένες τιμές του CN υπολογίζεται η μέγιστη δυνητική κατακράτηση για συντελεστή απωλειών 20% με βάση την παρακάτω εξίσωση:

$$S_{20} = 254(^{100}/_{CN_{20}} - 1)$$ Εξ. 7

Δεδομένου ότι ο συντελεστής απωλειών είναι πολύ μικρότερος από 20% στον ελλαδικό χώρο σύμφωνα με μελέτες που έχουν πραγματοποιηθεί, είναι απαραίτητη η διόρθωση του συντελεστή S σύμφωνα με την παρακάτω σχέση ώστε να ανταποκρίνεται σε συγκεκριμένο συντελεστή απωλειών:

$$S_\lambda = \frac{2\lambda h + (1-\lambda)h_e - \sqrt{h_e[h_e(1-\lambda)^2 + 4\lambda h]}}{2\lambda^2}$$ Εξ. 8

Στην συνέχεια, υπολογίζονται εκ νέου οι αρχικές απώλειες, η ενεργός βροχόπτωση καθώς και η τελική διορθωμένη τιμή του CN:

$$CN_a = 25400/(S_a + 254)$$ Εξ. 9

Λαμβάνοντας υπόψη το ψηφιακό μοντέλο εδάφους υπολογίζονται αρχικά οι επιφανειακές ταχύτητες με βάση την κλίση του εδάφους J και την παράμετρο k , συντελεστής τραχύτητας που προκύπτει από πίνακες σύμφωνα με τις χρήσεις γης κατά τον MacCuen.

$$V_o = k\, J^{1/2}$$ Εξ. 10

Σημειώνεται ότι αν η κλίση είναι μεγαλύτερη από 4%, για να αποφευχθεί υπερεκτίμηση της ταχύτητας γίνεται η παρακάτω διόρθωση:

$$J' = 0.05247 + 0.06363S - 0.182\, e^{-62.38J}$$ Εξ. 11

Αναφορικά με τις ταχύτητες στα διάφορα τμήματα του υδρογραφικού δικτύου αυτές υπολογίζονται με βάση την σχέση:

$$V_i = \beta_i\, J^{1/2}$$ Εξ. 12

Όπου η παράμετρος β θεωρείται σταθερά στο κάθε τμήμα και εξαρτάται από τα τοπικά χαρακτηριστικά του (τραχύτητα, γεωμετρία).

$$\beta_i = c_i/n_i$$ Εξ. 13

Ο παράγοντας ταχύτητας c εξαρτάται από το χρόνο συγκέντρωσης της λεκάνης t_c και άρα από το μήκος της μέγιστης διαδρομής του υδατορεύματος.

$$c = \left(\frac{L_{m1}}{\frac{\sqrt{J_{m1}}}{n_1}} + \frac{L_{m2}}{\frac{\sqrt{J_{m2}}}{n_2}} + \cdots + \frac{L_{mn}}{\frac{\sqrt{J_{mn}}}{n_n}} \right) t_r \quad \text{Εξ. 14}$$

Όπου t_r : ο χρόνος διαδρομής μέσω των τμημάτων του υδατορεύματος, υπολογιζόμενος μέσω της σχέσης

$$c = t_r = t_c - t_a \quad \text{Εξ. 15}$$

όπου t_a ο χρόνος συγκέντρωσης της υπολεκάνης ανάντη της κύριας διαδρομής του υδατορεύματος όπου συντελείται μόνο επίγεια ροή.

Στο πλαίσιο της παρούσας εργασίας ο χρόνος συγκέντρωσης είναι άμεσα εξαρτημένος από το επεισόδιο βροχής. Συνεπώς, η ακόλουθη σχέση εφαρμόζεται για τον υπολογισμό του χρόνου συγκέντρωσης:

$$t_{\mathrm{c}} = t_0 \, i_{\mathrm{e}}^{\,-\beta} \quad \text{Εξ. 16}$$

Ειδικά για το μοντέλο προσομοίωσης του ολικού υδρογραφήματος ισχύουν τα κάτωθι:

Λόγω του μικρού χρονικού ορίζοντα η εξατμισοδιαπνοή παραλείπεται από το ισοζύγιο νερού σε κάθε κελί. Η μέγιστη δυνητική κατακράτηση μεταβάλλεται χρονικά και συμβολίζεται ως S_t, και υποδεικνύει τηνχωρητικότητα μίας υποθετικής δεξαμενής ολικής χωρητικότητας K για εδαφική υγρασία. Η δεξαμενή προσομοιώνει την ακόρεστη ζώνη από την οποία η βροχόπτωση που διεισδύει μετατρέπεται σε υποδερμική ροή ή κατείσδυση σε βαθύτερες ζώνες. Η εξίσωση του υδατικού ισοζυγίου διαμορφώνεται ως:

$$W_t = W_{t-1} + I_t - Y_t - G_t \quad \text{Εξ. 17}$$

όπου W_t η εδαφικής υγρασία στο βήμα t, I_t η βροχή που διεισδύει, Y_t η υποδερμική ροή και G_t η κατείσδυση. Απαραίτητη παράμετρος είναι η αρχική εδαφική υγρασία W_0. Με την γνώση της μέγιστης δυνητικής κατακράτησης στην αρχή του επεισοδίου λόγω των προγενέστερων συνθηκών υγρασίας, η χωρητικότητα K υπολογίζεται ως:

$$K = W_0 + S_0 \quad \text{Εξ. 18}$$

Συνεπώς σε κάθε χρονικό βήμα με την επίλυση του ισοζυγίου η νέα μέγιστη δυνητική κατακράτηση βρίσκεται από την σχέση 19:

$$S_t = K - W_{t-1} \quad \text{Εξ. 19}$$

Η παραπάνω εξίσωση συνιστά μία σημαντική διαφοροποίηση από την κλασσική προσέγγιση SCS-CN που θεωρεί το μέγεθος S σταθερό στο χρόνο. Στην παρούσα προσέγγιση, το S μειώνεται όσο γεμίζει η δεξαμενή εδαφικής υγρασίας, δηλαδή η διήθηση βροχής είναι μεγαλύτερη από την κατείσδυση και τη δημιουργία υποδερμικής ροής. Αντίθετα, όταν σταματήσει η βροχόπτωση το S σταδιακά αυξάνεται. Η προσέγγιση αυτή επιτρέπει μεγαλύτερη ικανότητα αναπαράστασης μη-γραμμικών συμπεριφορών της διαδικασίας μετατροπής της βροχής σε απορροή και καλύτερη αναπαράσταση των πτωτικών κλάδων των υδρογραφημάτων.

Οι διεργασίες υποδερμικής ροής και κατείσδυσης περιγράφονται ως κλάσματα της εδαφικής υγρασίας σύμφωνα με τις σχέσεις:

$$Y_t = \kappa\, W_t \qquad \text{Εξ. 20}$$

$$G_t = \mu\, W_t \qquad \text{Εξ. 21}$$

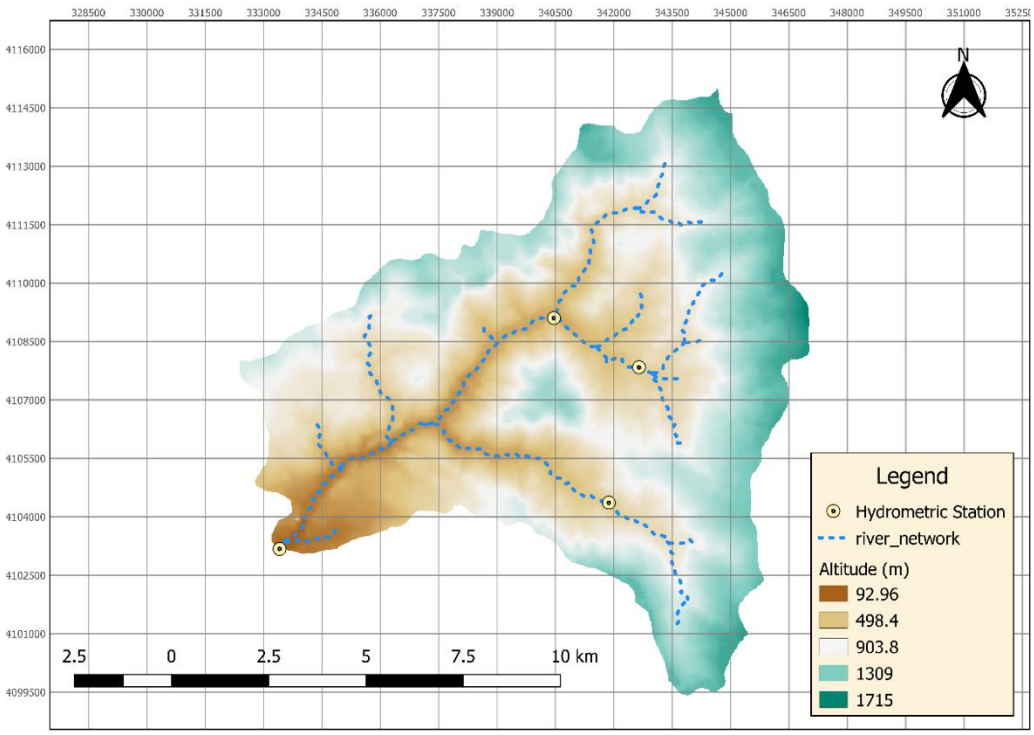
όπου κ και μ παράμετροι ποσοστού.

Και για τα δύο μοντέλα, για τη διόδευση της πλημμύρας στην έξοδο της λεκάνης απορροής χρησιμοποιείται η ευρέως διαδεδομένη μέθοδος των ισοχρόνων καμπυλών, αφού πρώτα εκτιμηθεί ο χρόνος διαδρομής κάθε φατνίου προς την έξοδο της λεκάνης, με βάσης τις ταχύτητες των φατνίων που συναντά στη διαδρομή (τύπου επίγειας ταχύτητας ή την ταχύτητα τμήματος υδρογραφικού δικτύου). Ειδικά για τη περίπτωση του μοντέλου ολικού υδρογραφήματος εισάγεται μία πρόσθετη παράμετρος

Περιοχή μελέτης

Ο ποταμός Νέδοντας ανήκει στο Υδατικό Διαμέρισμα Δυτικής Πελοποννήσου (GR01), και διέρχεται από την Καλαμάτα, πρωτεύουσας του Νομού Μεσσηνίας. Πηγάζει από τις δυτικές κλιτύες του Ταΰγετου και εκβάλλει στον Μεσσηνιακό Κόλπο, δυτικά του λιμανιού της Καλαμάτας, με συνολικό μήκος 26 km. Στη λεκάνη αυτή δεν εντοπίζονται έργα που θα μπορούσαν να μεταβάλλουν την υδρολογική της δίαιτα (π.χ. φράγματα, εκτροπές, λιμνοδεξαμενές). Τα μετεωρολογικά στοιχεία που συλλέγονται από σταθμό του αεροδρομίου στα 6 km δυτικά της Καλαμάτας παρέχουν πληροφορία για το υδατικό δυναμικό της περιοχής, το οποίο χαρακτηρίζεται από βροχοπτώσεις 600 mm στα νότια του Νομού Μεσσηνίας (Φοινικούντα – Μεθώνη), 1500 mm στα ορεινά και 800-1200 mm στις κεντρικές, βόρειες πεδινές και ημιορεινές περιοχές. Τα διαθέσιμα χωρικά δεδομένα, τα οποία χρησιμοποιήθηκαν για την λεκάνη απορροής, αφορούσαν την υψομετρική πληροφορία, δομημένη σε ένα δίκτυο τετραγωνικού πλέγματος ανάλυσης 25 m, τις χρήσεις γης (Corine 2000), το γεωλογικό υπόβαθρο και τις θέσεις χωροθέτησης των μετρητικών σταθμών. Στο Σχήμα που ακολουθεί παρουσιάζεται η εξεταζόμενη λεκάνη καθώς και το υδρογραφικό της δίκτυο.

Αξιοποιήθηκαν δύο καταγεγραμμένα επεισόδια με παρατηρήσεις βροχής σε διάφορους σταθμούς της περιοχής και παρατηρημένο υδρογράφημα στην θέση λατομείο Μπάκα.



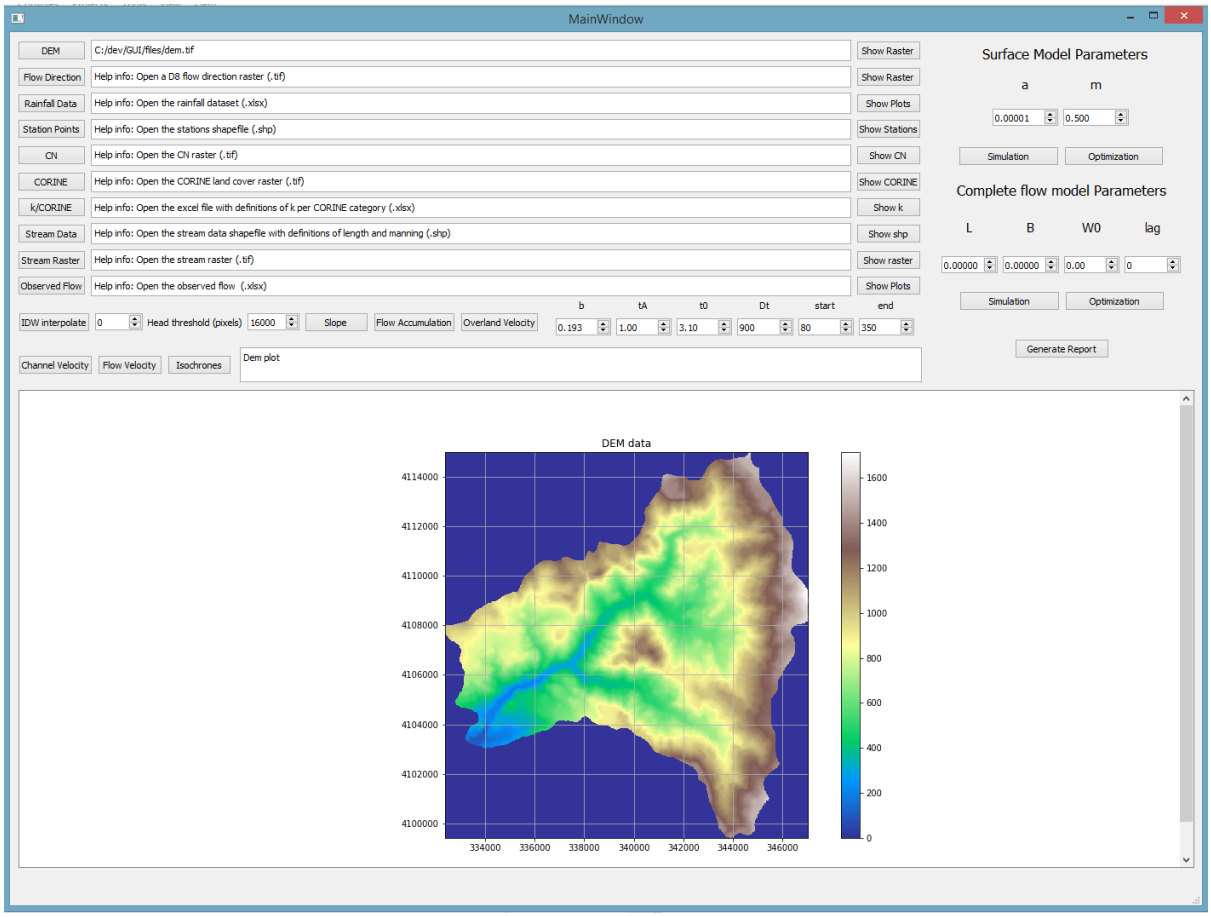
Σχήμα 1: Λεκάνη απορροής Νέδοντα και υδρογραφικό δίκτυο.

Ανάπτυξη λογισμικού

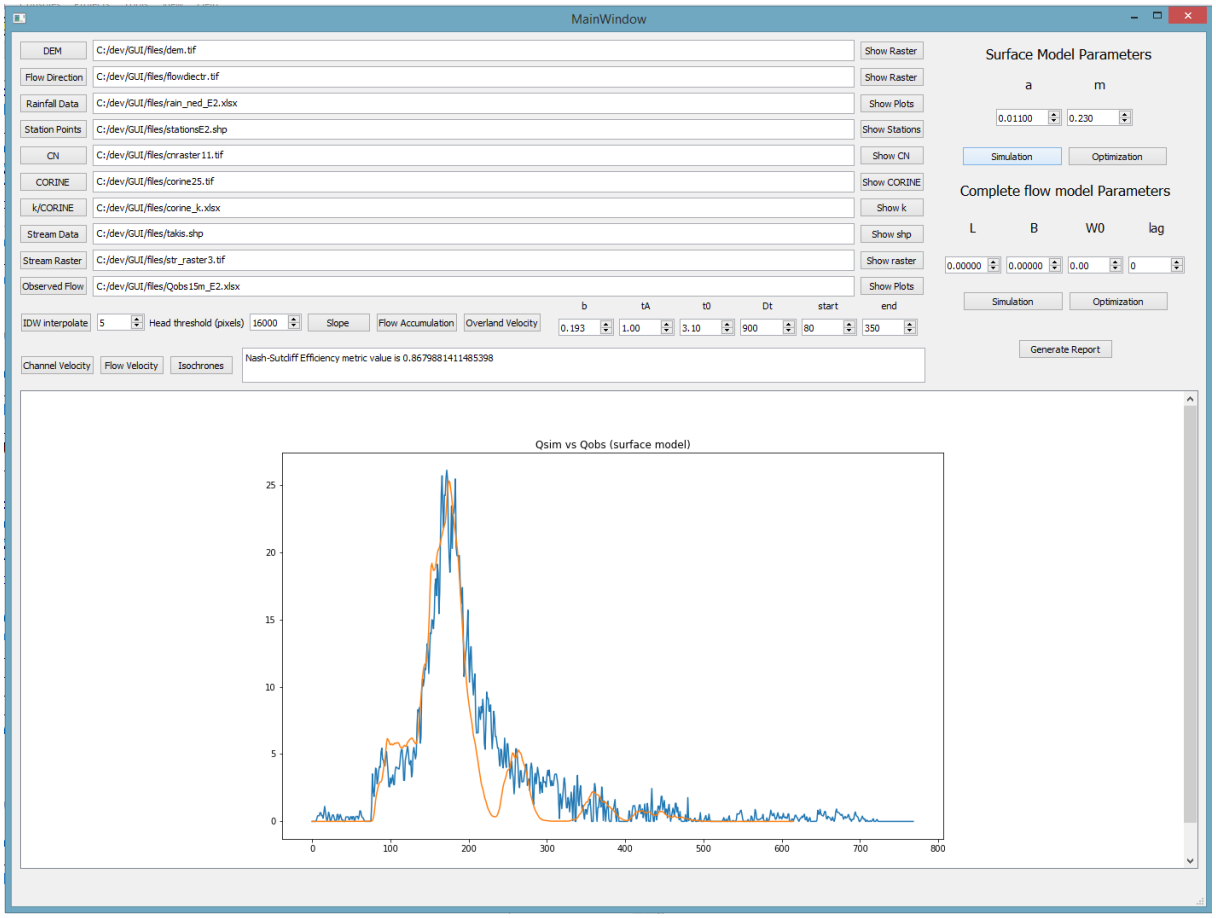
Για την εφαρμογή των παραπάνω μεθοδολογιών επιλέχθηκε η χρήση της γλώσσας Python, έναντι άλλων, κυρίως λόγω των πολλαπλών πλεονεκτημάτων της όπως για παράδειγμα το πλήθος πακέτων που παρέχει. Στην παρούσα εργασία, πέραν των δύο μοντέλων που γράφτηκαν στη συγκεκριμένη γλώσσα, δημιουργήθηκε και ένα λογισμικό για την καλύτερη διαχείριση των δεδομένων εισόδου, της αυτοματοποίησης των διαδικασιών προσομοίωσης και βελτιστοποίησης, καθώς και της οπτικοποίησης των εξαγόμενων αποτελεσμάτων. Στα Σχήματα που ακολουθούν παρουσιάζεται η μορφή του γενικού παραθύρου και διάφορων λειτουργιών.

Στο αριστερό τμήμα του παραθύρου ο χρήστης εισάγει όλα τα απαραίτητα δεδομένα. Αυτά είναι:

- Ψηφιακό Μοντέλο Εδάφους - DEM (.tiff)
- Κάνναβος διεύθυνσης ροής - Flow Direction (.tiff)
- Δεδομένα βροχόπτωσης (.xlsx)
- Βροχομετρικοί σταθμοί (.shp)
- Κάνναβος CN - Curve Number (.tiff)
- Κάνναβος χρήσεων γης (.tiff)
- Οι τιμές *k* με βάσει τις χρήσεις γης (.xlsx)
- Μήκος τμημάτων υδατορεύματος και τιμές Manning (.shp)
- Κάνναβος υδατορεύματος (.tiff)
- Παρατηρημένη απορροή στην έξοδο της λεκάνης (.xlsx)



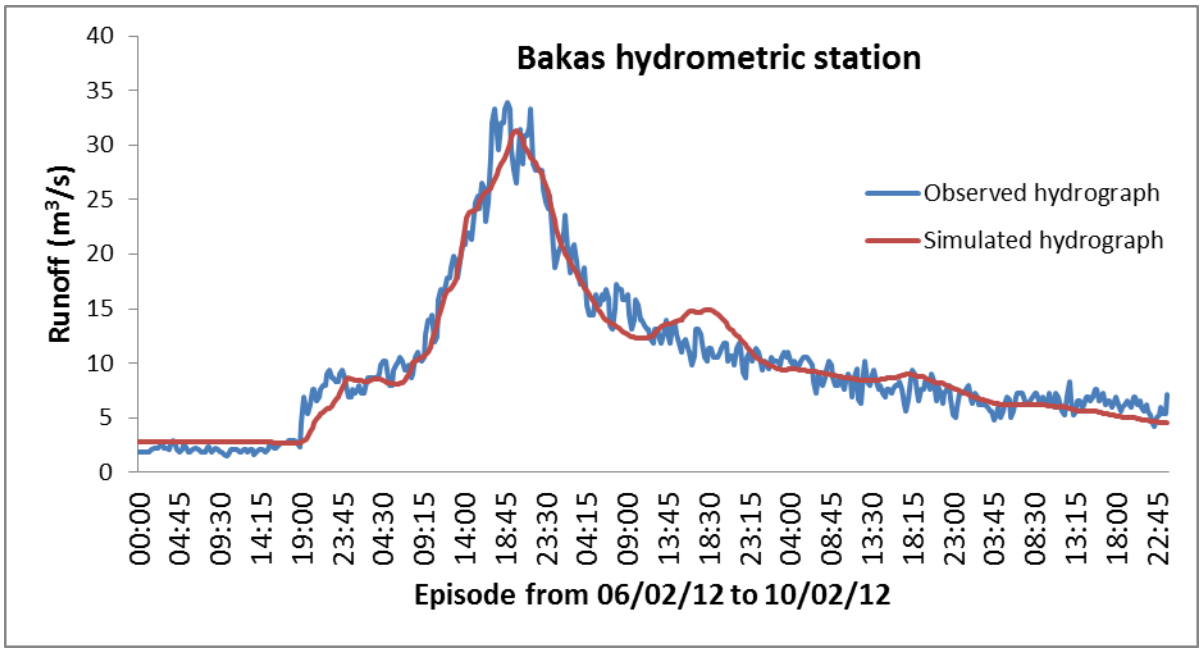
Σχήμα 2: Παράθυρο της εφαρμογής του λογισμικού, εισαγωγή και απεικόνιση DEM.



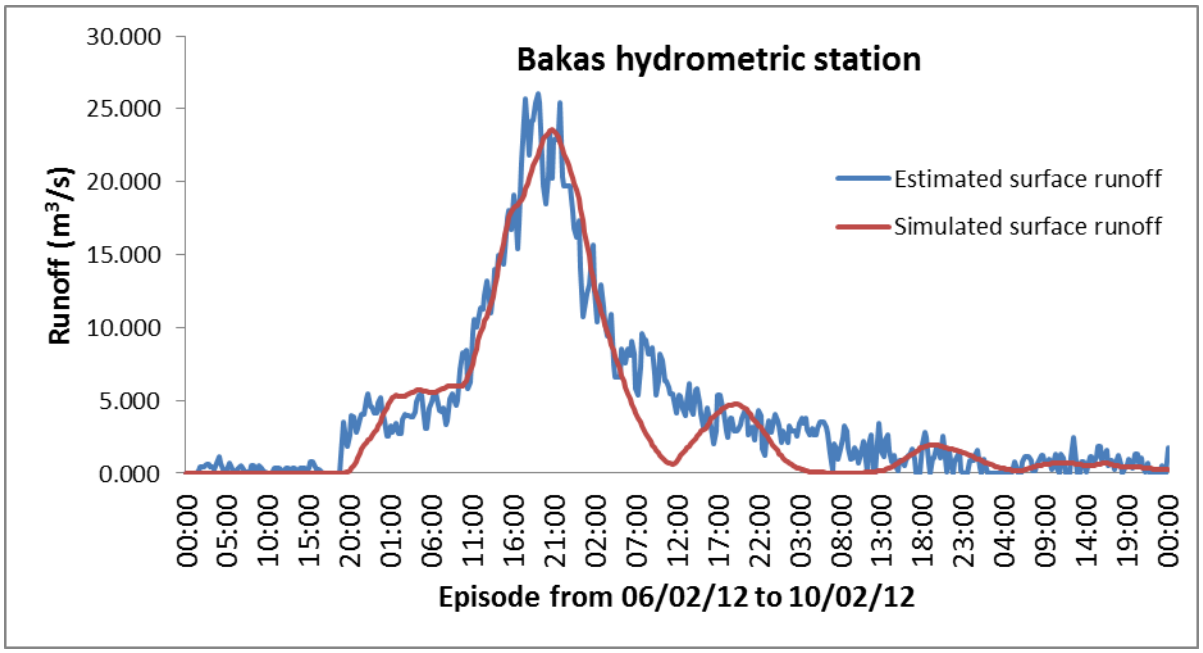
Σχήμα 3: Παράδειγμα διεξαγωγής προσομοίωσης με παραμέτρους ορισμένες από το χρήστη.

Αποτελέσματα εφαρμογής μοντέλων

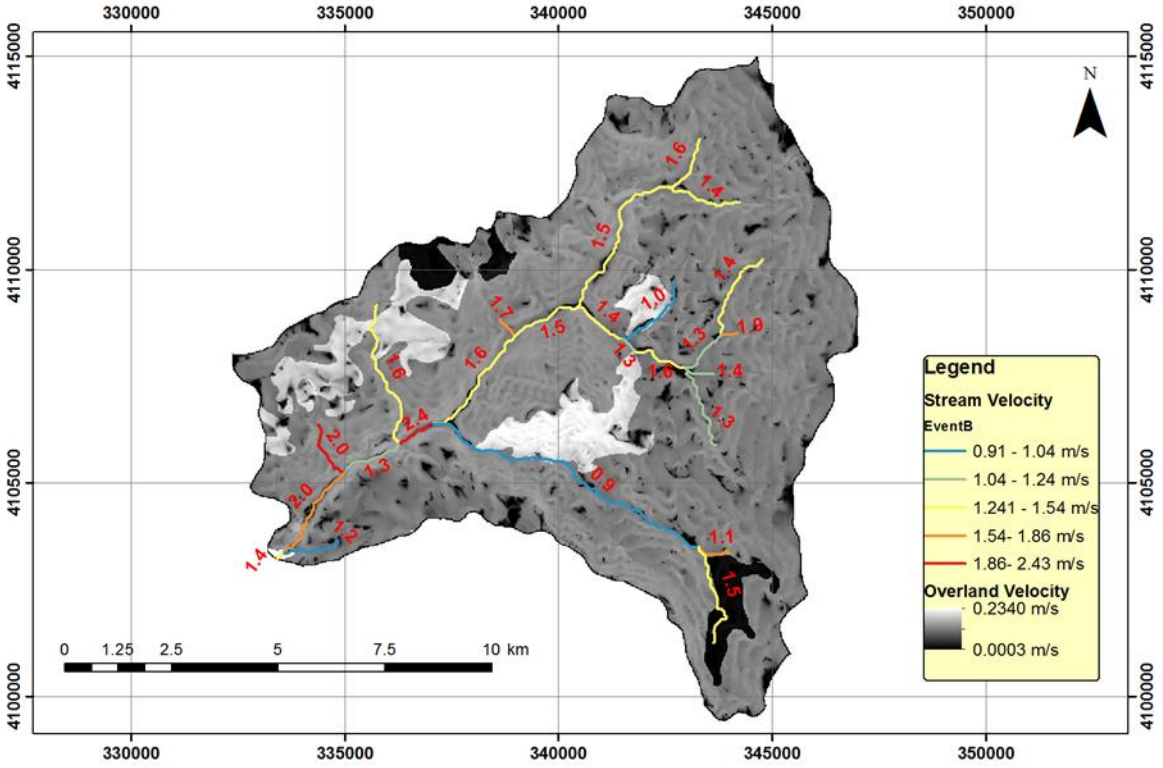
Στα σχήματα 4 και 5 παρουσιάζονται δύο παραδείγματα από την επιτυχή εφαρμογή των δύο κατανεμημένων μοντέλων που αναπτύχθηκαν στην παρούσα εργασία (συντελεστές Nash-Sutcliffe Efficiency 0.95 και 0.90 αντίστοιχα) Συνολικά, εξετάστηκαν δύο διαφορετικά επεισόδια βροχόπτωσης, και οι συντελεστές απόδοσης Nash-Sutcliffe και των δύο μοντέλων ήταν ιδιαίτερα καλοί, ενώ και η αναπαραγωγή των αιχμών και των χρόνων εμφάνισης τους χαρακτηρίζονται από αξιοπιστία. Στο Σχήμα 6 απεικονίζεται ως παράδειγμα η χωρική κατανομή της επιφανειακής ταχύτητας του νερού καθώς και η χωρική κατανομή της ταχύτητας στο υδρογραφικό δίκτυο για ένα από τα επεισόδια.



Σχήμα 4: Σύγκριση μοντέλου συνολικού υδρογραφήματος με παρατηρημένο υδρογράφημα.



Σχήμα 5: Σύγκριση μοντέλου επιφανειακής απορροής με παρατηρημένο υδρογράφημα όπου έχει αφαιρεθεί η υποδερμική ροή.



Σχήμα 6: Ταχύτητες κελιών του κατανεμημένου μοντέλου (δύο τύποι, επιφανειακή ταχύτητα και ταχύτητες κελιών που ανήκουν σε κλάδους του υδρογραφικού δικτύου).

Συμπεράσματα

Η εργασία αυτή είχε δύο στόχους, ένα ερευνητικό και έναν τεχνολογικό.Ως προς τον ερευνητικό της σκοπό, καταστρώθηκαν δύο κατανεμημένα μοντέλα τα οποία χαρακτηρίζονται από πολύ καλές επιδόσεις στην εφαρμογή τους στην δεδομένη περιοχή μελέτης. Στη μεθοδολογία που αναπτύχθηκε συμβάλουν οι παρακάτω καινοτομίες:

- Ενσωματώνεται μία σύγχρονη μέθοδος GIS για την εκτίμηση του δείκτη CN από γεωχωρική πληροφορία.
- Αξιοποιείται μία εμπειρική σχέση για την αναγωγή του CN σε οποιασδήποτε συνθήκες υγρασίας πριν το συμβάν βροχόπτωσης.
- Εισάγεται μία μέθοδος αναγωγής του CN για οποιαδήποτε αρχική κατακράτηση.
- Ενσωματώνεται σε κατανεμημένο μοντέλο η λογική του μεταβαλλόμενου με την ένταση της απορροής χρόνου συγκέντρωσης.
- Διορθώνονται χωρικά οι ταχύτητες στους κλάδους του υδρολογικού δικτύου ανάλογα με τα φυσιογραφικά χαρακτηριστικά τους και δίνεται η δυνατότητα για διόδευση με ικανοποιητική ακρίβεια χωρίς τη χρήση υδραυλικών μοντέλων.
- Προσθήκη εξισώσεων ισοζυγίου στις σχέσεις της μεθόδου SCS για την προσομοίωση της υποδερμικής ροής και της μεταβαλλόμενης χρονικά μέγιστης δυνητικής κατακράτησης
- Φειδωλή προσέγγιση, με λίγες παραμέτρους, σε αντίθεση με πολλά άλλα κατανεμημένα μοντέλα που χρειάζονται πλήθος παραμέτρων για την λειτουργία τους.

Ως προς τον τεχνολογικό σκοπό της η εργασία φέρνει σε σύζευξη πολλά υπολογιστικά και προγραμματιστικά εργαλεία, ανοιχτού κώδικα, τα οποία είναι ιδιαίτερα χρήσιμα στον σύγχρονο υδρολόγο-μηχανικό, για ποικίλες χρήσεις: γεωχωρική ανάλυση, στατιστική,

υπολογιστικά πακέτα, αλγόριθμοι βελτιστοποίησης, διαχείριση δεδομένων, οπτικοποίηση αποτελεσμάτων κ.α., τα οποία διασυνδέονται μέσω της καταξιωμένης στον επιστημονικό κόσμο γλώσσας Python. Στο πλαίσιο της παρούσας εργασίας αναπτύχθηκε μαζί με τα υδρολογικά μοντέλα και ένα εύχρηστο λογισμικό για την αυτοματοποίηση όλων των διαδικασιών.

Λέξεις – Κλειδιά: μοντέλα βροχής-απορροής, Python & υδρολογία, βελτιωμένη NRCS-CN μέθοδος, ταχύτητα υδατορεύματος εξαρτώμενη από ένταση απορροής, κατανεμημένα μοντέλα μεμονωμένων επεισοδίων.

Figure Catalogue

Table Catalogue

TABLE OF CONTENTS

1 Introduction

1.1 General context

The scope of this study is the development of an event based distributed hydrological model dealing with rainfall-runoff simulation. In contrast with most contemporary hydrological models (some of them discussed in Chapter 2), we seek for a parsimonious approach in data requirements and parameters needed. Also, this study aims to incorporate many novel and recent methodologies/techniques that fuse geospatial operations with rainfall – runoff modeling and attempt to solve a common problem with most rainfall-runoff models that are not coupled with hydraulic models: that of the oversimplified assumption of a spatiotemporally constant value of channel network velocity.

Also, there is a second, but equally important HydroInformatics – related objective of this thesis: the integration of various tools that help the modern hydrologist perform data analysis, geospatial operations, simulation/optimization, numerical analysis and visualization in a single platform. For automation of data handling, simulation and optimization procedures and visualization, a GUI software is implemented, written in the high-level programming language Python. Also, novel programming paradigms (in the sense of the usual "engineering" approach to programming) such as parallel programming and Just-In-Time (JIT) compilation are implemented, so as to reduce the typically expensive computation cost (in terms of execution time) associated with distributed models that operate in fine time and spatial scales (usually time step smaller than one hour, and grid size smaller than 100 m) These fine scales exponentially increase computations needed and such models are of particular interest for code optimization.

1.2 Structure of the thesis

This thesis comprises ten chapters:

In the **second Chapter** a literature review is conducted in order to document and analyze the theoretical background regarding the rainfall – runoff models. Also, some of the most known distributed models are described.

The **third Chapter** is a review of GIS programming tools, presented in order to illustrate the most common available tools in hydrological studies and to stimulate the reader about modern practices.

The **fourth Chapter** consists of an overview of the chosen programming language, Python, and its advantages over others. In addition, the packages and libraries employed in this thesis are introduced.

In **Chapter five** the applied methodology of this study is thoroughly described. All the procedures used in the implementation of the two models and the novelties of this work are listed in this chapter.

Chapter six provides the necessary information regarding the calibration procedure.

Chapter seven is the model implementation in Python, in which the procedure followed for the creation of the two models is described in detail. Also, in the same chapter is described the software application which is developed, with all its capabilities.

In **Chapter 8**, Nedontas stream basin, which is the study case of this thesis is presented and the relevant data.

Next, **Chapter 9** contains the results, the parameters as well as the performance metrics of the two models implemented.

Chapter 10 discusses the conclusions of this study. This chapter consists of a summary of the topics treated in this thesis, the novelties of the work, final remarks regarding the conclusions made from all analyses and, finally, suggestions for further research.

2 Review of distributed hydrological models

As the scope of this work is the development of an event-based distributed hydrological model, this chapter acts as a brief introduction to rainfall-runoff modelling and a short review of some widespread distributed models that are extensively referred in the literature.

2.1 Classification of rainfall-runoff models

Rainfall – runoff models are no different than other general computational models in the sense that they are simplified representations of real-world systems. In fact, hydrological models deal with the complex nature of rainfall-runoff processes, which is determined by a number of highly interconnected water, energy and vegetation processes at various and mixed spatial scales, most of them not well understood or easy to represent with equations and/or models. All these processes exhibit large uncertainty, thus making essential to employ several modeling assumptions. Hydrologists rely on their self- understanding of the system gained through interaction with it, observation and experiments, in what is known as perceptual modeling (Beven, 2001).

There are many classifications of rainfall-runoff models found in the literature. The most common distinctions are between (Sorooshian *et al.*, 2008):

- Deterministic and stochastic models: Deterministic models generate the same output for every model run, for given parameters, whereas stochastic models use probabilistic inputs, thus providing different outputs for given parameters.
- Event-based and continuous: Event-based models run for a specific time period, to represent the response of the basin against a given rainfall event, whereas continuous models simulate arbitrary long periods of time, comprising both rainfall and non-rainfall events and generally changing conditions throughout simulation. Typically, event-based models are quite sensitive against the initial conditions, which should be carefully determined by the user.
- Physically-based (white-box), empirical (black-box) and conceptual (grey-box) models: The differences of these three approaches are summarized in Table 2.1.

Table 2.1: Summary of empirical, conceptual and physically-based models (adapted from Gayathri *et al.*, 2015).

Empirical	Conceptual	Physically-based
• Data-driven or metric or black box • Involve mathematical equations, derive value from available time series • Little consideration of features and processes of the system	• Parametric or grey box • Based on modeling of reservoirs and include semi-empirical equations with macroscopic physical basis • Parameters are derived from calibration	• Mechanistic or white box • Based on spatial distribution, evaluation of parameters describing physical characteristics • Require data about initial state of model and morphology of the catchment

• High predictive power, low explanatory depth • Cannot be transferred to other catchments • Examples: ANN, unit hydrograph • Valid within the boundary of given domain	• Simple and easily implemented in computer code • Require large hydrological and meteorological data • Examples: HBV model, TOPMODEL • Calibration involves curve fitting make difficult physical interpretation	• Complex model, requiring human expertise and computational capability • Suffer from scale-related problems • Examples: SHE/MIKESHE model, SWAT • Valid for wide range of situations

- Lumped and distributed models:

The latter is essentially the most distinctive model classification. In lumped models, the spatial variability is discarded from the model, as the whole basin is a unit generating runoff. Thus, the input data (mainly the precipitation) is forced to relate with the output data (streamflow) without considering spatial processes, patterns and characteristics. The lumped hydrologic models impose many assumptions, especially in large watersheds, as variables and parameters are representative average values (lumped) for a river basin with semi – empirical equations describing the physics (Refsgaard, 1996). Lumped models were first conceived nearly 170 years ago, to address the limitation of data and computational power. The first widely used rainfall – runoff model is attributed to Mulvaney (1851), and it is widely known as the rational method.

On the other hand, a distributed model accounts for spatial variations of processes and properties, thereby explicit characterization of the processes and patterns is made (Beven, 1985; Refsgaard, 1996; Smith *et al.*, 2004). Probably, the first distributed model was introduced by Ross (1921), who attempted to divide zones in the catchment area on the bases of travel time to outlet and used routing techniques. The modern availability of high spatial resolution data such as DEM, precipitation, vegetation, soil and other atmospheric variables has led to a recent surge in developing many sophisticated distributed hydrologic models. In a distributed physically –based model the water and energy fluxes are usually computed from the prevailing partial differential equations (e.g., Saint Venant's equations for overland and channel flow, Richard's equation for unsaturated flow and Boussinesq's equation for groundwater flow) (Refsgaard, 1996). Distributed models have advantages, in terms of considering spatially variable inputs and outputs, assessment of pollutants and sediment transport, and also analyzing the hydrological responses at ungauged basins (Smith *et al.*, 2004). Due to the fact that distributed models make distributed predictions, there is a lot of potential for evaluating not only the predictions of discharge at a catchment outlet, but also the internal state variables, such as water table levels, soil moisture levels, and channel flows at different points on the network. In the last two decades, few attempts were made to validate the predictions. This is clearly partly due to the difficulty of collecting measurements (Beven, 2012). It is worth noting that all distributed models require effective parameter values to be specified at the scale of the calculation elements that may be different from values measured in the field. Distributed predictions indicate that distributed data can be used in model calibration but evaluation of this type of model may be difficult due to differences in scale of predictions and measurements, and

the fact that the initial and boundary conditions for the model cannot be specified with sufficient accuracy.

2.2 Physical processes modeled within rainfall-runoff models

Beven (2012) argues that our understanding on the physical processes is still evolving and there is a debate among many hydrologists about the most important processes in rainfall-runoff modelling and generally, different processes may be dominant in different environments. Conceptually, most hydrological models incorporate variations of the processes represented in Figure 2.1. These are mostly attributed to mechanisms from the seminal works of Horton (1933), Cappus (1960), Hewlett (1961), Hursch (1936), Betson (1964), Dunne (1970), and Weyman (1970), as depicted in Figure 2.2 (Beven, 2012).

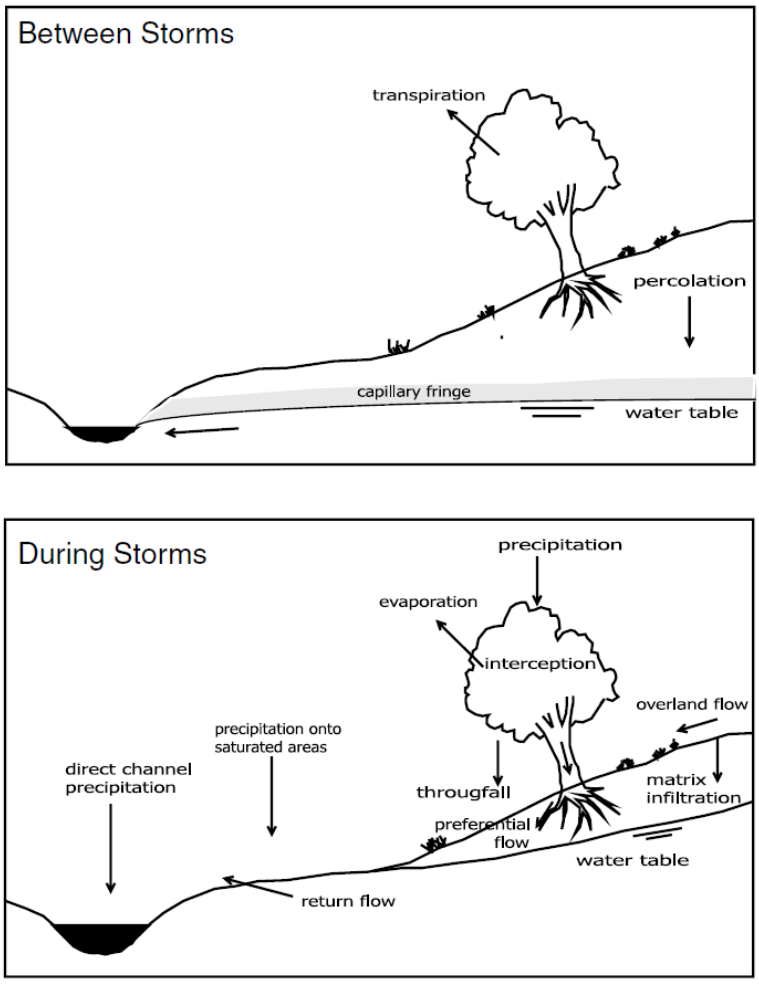


Figure 2.1: Hydrological processes incorporated in most rainfall-runoff models (Beven, 1991).

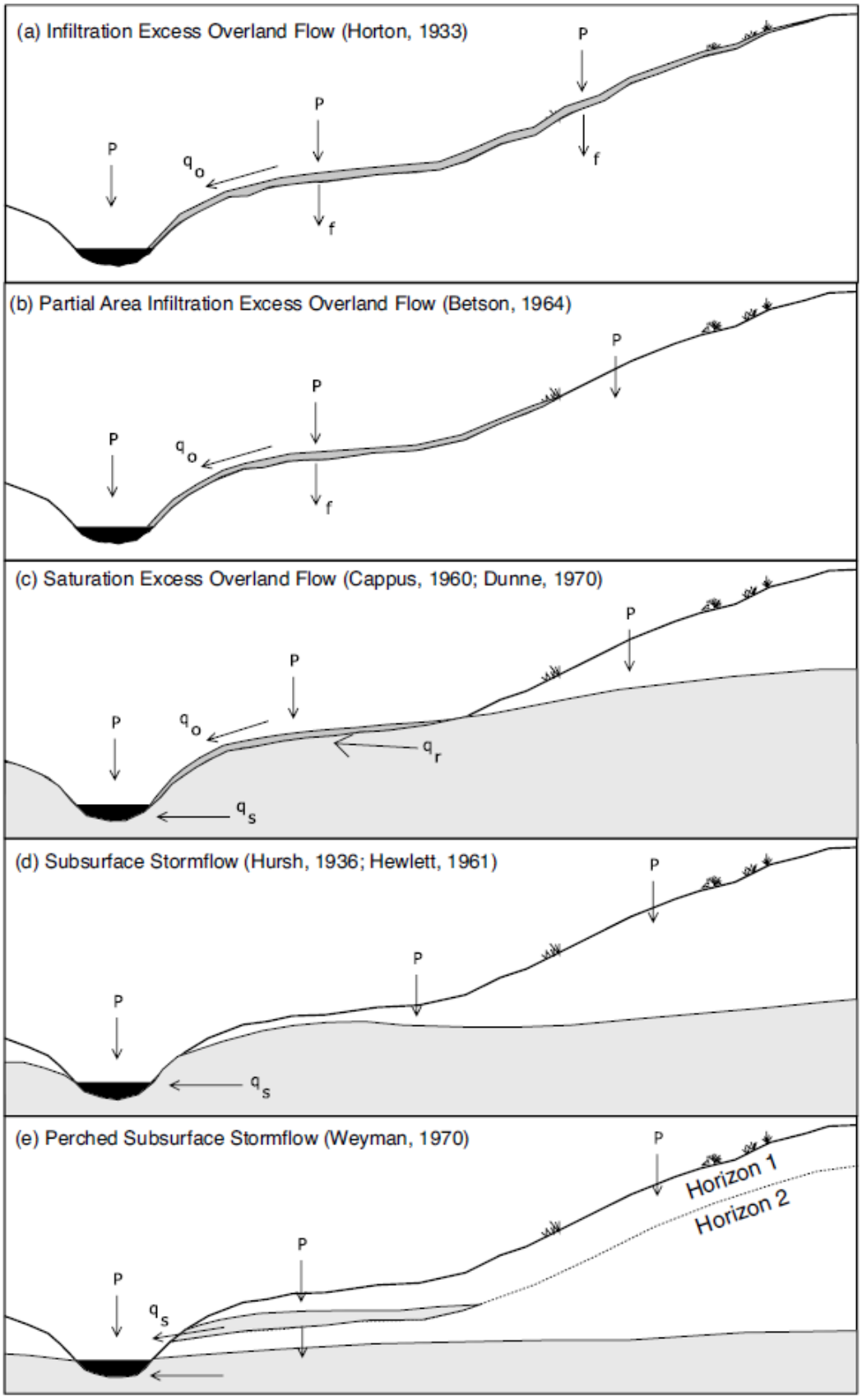


Figure 2.2: Mechanisms represented in seminal works in hydrology (Beven, 2012).

In their simpler form, hydrological models require two essential components: one to determine how much of a rainfall event becomes part of the storm hydrograph (the runoff generation component, related to volume), the other to take account of the distribution of that runoff in time to form the shape of the storm hydrograph (the runoff routing component, related to temporal distribution). These two components may appear in many different guises and degrees of complexity in different models, but they are always there in any rainfall–runoff model, together with the difficulty of clearly separating one component from the other. In general, it is accepted that the runoff generation problem is the more difficult (Beven, 2012). Practical experience suggests that the complexities and nonlinearities of the flow generation processes are much greater than for the routing processes, and that relatively simple routing models may suffice. One of the most common routing processes in the literature is the Muskingum method.

2.3 Brief review of common distributed hydrological models

Some hydrological models based on a grid to grid approach are listed below. Firstly, the fully 3-D models of Binley *et al.* (1989a, 1989b) and Paniconi and Wood (1993) use a grid-based spatial discretization. The ANSWERS model (Beasley *et al.*, 1980; Silburn and Connolly, 1995; Connolly *et al.*, 1997), which has its origins in one of the very first fully distributed grid-based models of Huggins and Monke (1968), essentially considers only an infiltration excess runoff generation mechanism, using the Green–Ampt infiltration equation to predict excess rainfall on each grid element. The runoff generated is then routed towards the stream channel in the direction of steepest descent from each grid element. The CASC2D model of Doe *et al.* (1996) and Downer *et al.* (2002) is similar in that it also uses a Green–Ampt infiltration equation, but it uses a 2-D diffusion wave approximation to model overland flow on the hillslopes and a 1-D diffusion wave model for the channel reaches. CASC2D was later extended to include more subsurface flow processes as the Gridded Surface/Subsurface Hydrologic Analysis (GSSHA) model (Downer and Ogden, 2004; Downer *et al.*, 2005). Moreover, the 3-D version of HILLFLOW of Bronstert and Plate (1997) is a grid – based model, with the interesting option of modelling the Richards equation using the *fuzzy logic* methodology of Bardossy *et al.* (1995). HILLFLOW also has a 2-D option for modelling individual hillslope elements.

2.3.1 SHE Model

A widely known hydrological model based on grid elements is the Système Hydrologique Européen (SHE) model, which was introduced in 1977, as collaboration between the UK Institute of Hydrology, the Danish Hydraulics Institute (DHI) and SOGREAH of Grenoble in France. Beven *et al.* (1980) have published an early description of the model. Later, an explanation of the modelling philosophy was provided by Abbott *et al.* (1986a, 1986b), and finally, the first full application, to the Institute of Hydrology River Wye experimental catchments at Plynlimon, Wales (10 km^2) was published in a series of articles by Bathurst (1986a, b).

SHE, which is a grid – based model, divides the catchment into a number of square or rectangular grid elements, linked to channel reaches that run along the boundaries of the hillslope grid. The grid size in case studies that employ SHE ranges from 50 m up to 2 km in a recent case study for the Kolar and Narmada catchments in India, which is a very wide range. However, it is obvious that in the latter case the grid size is so large that the model cannot be considered to be representing flow on the hillslopes or in the smaller channels of the catchment in any meaningful way.

Each hillslope grid element has a specified surface elevation and model components for interception, evapotranspiration, snowmelt and one-dimensional vertical unsaturated zone flow where appropriate. A two-dimensional surface runoff and groundwater components links the

grid elements. Internal boundary conditions allow the coupling of surface flow and infiltration into the unsaturated zone, the unsaturated and saturated zones at the local water table, and groundwater and channel flows. The model is able to predict a variety of runoff generation processes on each grid element, including both infiltration excess and saturation excess runoff, and the groundwater flow component can be used to simulate subsurface contributions to the hydrograph under suitable conditions.

The main disadvantage of this model is the large number of parameters. The parameter values required are effective values at the grid element scale, which may not be the same as values that might be measured locally. Nevertheless, the model also offers the potential to specify fully distributed precipitation and meteorological data across the model grid elements, if the data are available. The predictions are, however, dependent on the grid scale used.

2.3.2 SHE evolution

A UK version of SHE is SHETRAN, which is based within the Water Resource Systems Research Unit at the University of Newcastle. SHETRAN has added contaminant and sediment transport components (Bathurst *et al.*, 1995, 2004; Ewen *et al.* 2000). The DHI version, MIKE SHE, has also added a contaminant transport component (Refsgaard and Storm, 1995). In both cases, the predictions of contaminant transport are based on the advection–dispersion equation. Both DHI and the University of Newcastle now have versions of SHE which make fully 3 –D solutions for the unsaturated – saturated flow domain. MIKE SHE has also added options to use a simple groundwater store where a fully subsurface solution is not justified and to predict a preferential recharge to the saturated zone as a simple proportion of the infiltration rate (Refsgaard and Storm, 1995). Such modifications undermine the way in which models purport to be “physically-based”.

2.3.3 G2G model

In the Probability Distributed Moisture PDM model, which is described in Figure 2.3, the multiple storage elements are allowed to fill and drain during rainstorm and interstorm periods respectively. If any storage component is full then any additional rainfall is assumed to reach the channel quickly as storm runoff. A slow drainage component is allowed to deplete the storages between storms, contributing to the recession discharge in the channel and setting up the initial storages prior to the next storm. Evapotranspiration is also taken from each store element during the interstorm periods.

The advantages of the PDM model are its analytical and computational simplicity. It has been shown to provide good simulations of observed discharges in many applications so that the distribution of conceptual storages can be interpreted as a realistic representation of the functioning of the catchment in terms of runoff generation. However, no further interpretation in terms of the pattern of responses is possible, since there is no way of assigning particular locations to the storage elements. In this sense, the PDM model remains a lumped representation at the catchment scale.

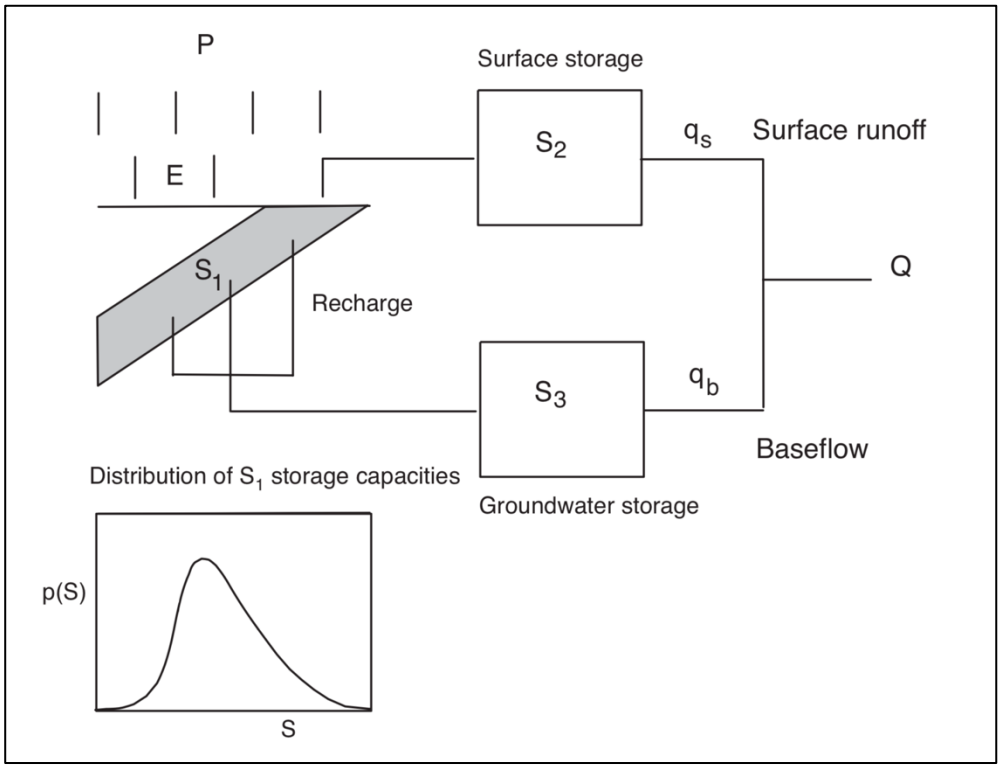


Figure 2.3: Structure of the PDM model (K.J. Beven, 2012).

The Figure 2.4 illustrates a PDM model, used as a semi-distributed model, where PDM elements represent grid squares feeding a grid-to-grid routing method.

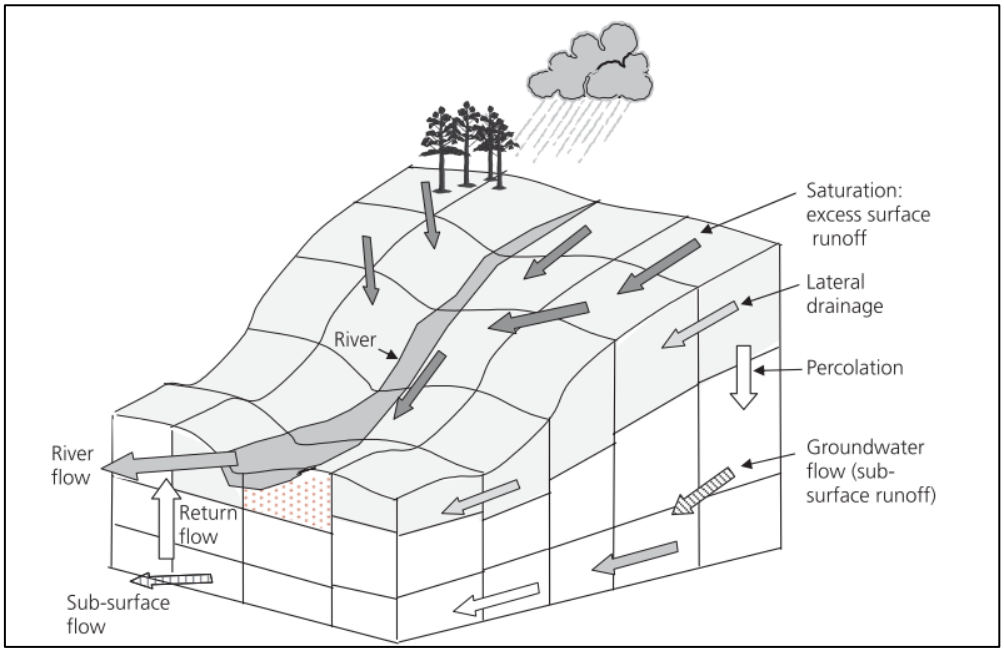


Figure 2.4: Integration of the PDM grid elements inti the G2G model (after Moore *et al.* 2006).

This Grid to Grid (G2G) model is directly descended from the distributed forms of PDM model noted above and makes use of the same way of relating the maximum storage in the runoff generation function to the local mean slope for each grid.

The G2G model is the first example of a hydrological model for the whole of the UK. It makes use of calibration, wherever gauging station data are available, but can also provide predictions for ungauged subcatchments and catchments, if data are not available. Any grid element in the country can be interpreted (or color coded in a visualization) for the current state of the flow in either absolute terms or as a frequency of occurrence. This makes the model a useful tool for forecasting purposes.

G2G has now operational use in the UK (Met Office Global and Regional Ensemble Prediction System - MOGREPS), since it provides predictions of potential flooding with long lead times. This model runs on a 1 km grid for the whole of the UK up to five days ahead, enabling a probabilistic evaluation of the potential for flooding across the country.

2.3.4 TOPMODEL

A simple approach to predicting spatial patterns of the responses in a catchment is represented by TOPMODEL (Beven *et al.* 1995; Beven 1997). This aims to develop a pragmatic and practical forecasting and continuous simulation model and a theoretical framework within which perceived hydrological processes and model procedures may be researched.

The parameters of the model are physically interpretable and few in number, so as to ensure that values determined by a calibration exercise should be more easily identifiable. This model represents an attempt to combine the computational and parametric efficiency of a distribution function approach, with the link to physical theory and possibilities for more rigorous evaluation of spatial patterns of predictions offered by a fully distributed model.

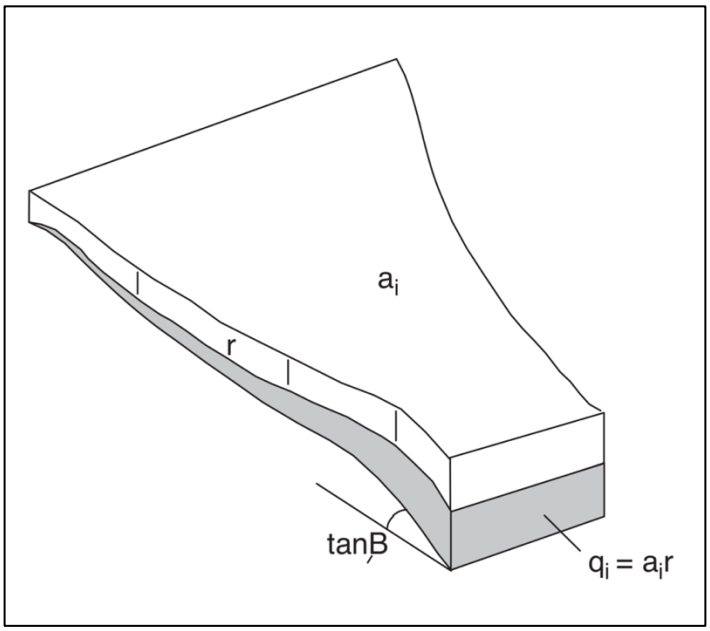


Figure 2.5: Definition of the upslope area and draining through a point within a catchment (K. J. Beven, 2006).

TOPMODEL is considered as an improved approximation of the kinematic wave description of the subsurface flow system. This link was explicitly made by Kirkby (1997) and Wigmosta and Lettenmaier (1999). It is premised upon two basic assumptions: that the dynamics of the saturated zone can be approximated by successive steady state representations of the saturated zone on an area a draining to a point on a hillslope (Figure 2.3) and that the hydraulic gradient

of the saturated zone can be approximated by the local surface topographic slope, tan β. These assumptions lead to simple relationships between catchment storage (or storage deficit below saturation), in which the main factor is the Kirkby topographic index (a/ tan β) (Kirkby, 1975), which represents the propensity of any location to reach saturated conditions.

TOPMODEL was initially developed for simulating small catchments in the UK. The results indicate that reasonable results are possible using the minimum parameters for calibration. It is worth mentioning that catchments with deeper groundwater systems or locally perched saturated zones are more difficult to model. These tend to go through a wetting up sequence at the end of the summer period in which the controls on recharge to any saturated zones and the connectivity of local saturated zones may change with time. Durand *et al.* (1992) have shown that this model can successfully simulate discharges in catchments with fast responses.

Regarding the software involved with TOPMODEL, there are two programs associated. The first one, for initial analysis of a catchment DTM (DTMAnalysis) and the second one TOPMODEL99 to simulate hydrographs and contributing areas, but also to carry out model sensitivity analysis.

Overall, this rainfall – runoff model, by employing an index of hydrological similarity using topography and soil information, can map the predictions. The calculations are based on the distribution of the index, which greatly reduces the required computer resources. The TOPMODEL concepts are not, however, applicable everywhere, particularly in catchments subject to strong seasonal drying when the basic assumptions underlying the index break down.

It is important to mention that the simplicity of the TOPMODEL calculations have allowed the interaction between grid resolution of the topographic analysis and calibrated parameter values to be studied in a number of applications. A similar interaction between scale of discretization and effective parameter values should hold for more complex models, including physically based fully distributed models, but may not be so readily apparent.

2.3.5 SWAT model (Soil and Water Assessment Tool)

The development of SWAT is an ongoing procedure and it is the successor of "the Simulator for Water Resources in Rural Basins" model (SWRRB). SWAT model is a complex physically – based model and was designed to test and forecast the water and sediment circulation and agriculture production with chemicals in ungauged basins. It is efficient in performing long-term simulations. The model delineates the entire catchment into sub-catchments, which are further divided into hydrologic response units (HRU), on the basis of land use, vegetation and soil characteristics. Model inputs are daily rainfall data, maximum and minimum air temperature, solar radiation, relative air humidity and wind speed, while its outputs are water and sediment fluxes, vegetation growth and nutrients concentrations. Snowfall is estimated on the basis of precipitation and mean daily air temperature, while the methods of Penman-Monteith, Priestly-Taylor and Hargreaves are used for the estimation of potential evapotranspiration. In order to obtain accurate forecasting of water, nutrient and sediment fluxes, it is necessary to simulate the full hydrologic cycle of the catchment, by employing the water balance equation:

$$SW_t = SW_o + \sum_{i=1}^{t} R_v - Q_s - W_s - ET - Q_{gs} \quad \textbf{(2.1)}$$

where SW_t is the humidity of soil, SW_0 is base humidity, R_v is rainfall volume in mm water, Q_s is the surface runoff, W_s is seepage of water from soil to underlying layers, ET is evapotranspiration, Q_{gs} is ground water runoff and t is time, in days.

3 Computational tools for GIS programming

Considering the spatial character of parameters and inputs controlling the hydrological processes across a river basin, it is not surprising that Geographic Information Systems (GIS) have become an integral part of hydrological studies. This chapter provides an overview of open source GIS software used in hydrological and geospatial analysis, used in the course of this work and providing insight on useful functionality. As a remark, in the early stages of the thesis, the proposed model in chapter 7 was developed as a QGIS/Grass plugin, before implementing it as a standalone application.

3.1 QGIS

3.1.1 General

QGIS is an Open Source Geographic Information System (GIS) that was established in 2002, by G. Sherman. The project was incubated with the Open Source Geospatial Foundation (OSGeo) in 2007. Initially, it aimed to provide a GIS data viewer for common geospatial formats, however functionality over time increased exponentially due to a large community of users and developers. It has reached a point in its evolution where it is being used for daily GIS data analysis in many fields replacing paid professional GIS services. This Open Source GIS could be installed on Windows, Mac OS X, Unix, Linux, and Android operating systems, making it a very flexible software package, which any scientist could use and develop.

3.1.2 Overview of QGIS

This section provides an overview of the basic functionalities of the program, and its use in the domain of hydrology, hydraulics and water resources management.

QGIS is composed of two programs, QGIS Desktop and QGIS Browser. Desktop is used for managing, displaying, analyzing, and styling data, while the Browser is used to manage and preview data. One of the main strengths of this program is its ability to load a large number of data types. The user is able to load vector files, with the opportunity to choose the source type and source of the dataset. The commonly used flat file types are ESRI shapefile (.shp), AutoCAD DXF (.dxf), Comma separated values (.csv), GPS eXchange Format (.gpx), Keyhole Markup Language (.kml), SQLite/SpatiaLite (.sqlite/.db). QGIS Directory can load data stored on disk that is encased in a directory. The commonly used directory types are U.S. Foundation. As a library, it presents a single raster abstract data model and single vector abstract data model to the calling application for all supported formats. It also comes with a variety of useful command line utilities for data translation and processing. QGIS, as well as many other programs, use GDAL to handle many geospatial data processing tasks. QGIS supports PostGIS, SpatiaLite, MSSQL, and Oracle databases.

Also, QGIS supports the loading of OGC-compliant web services, such as WMS/WMTS, WCS, and WFS. Loading a web service is similar to loading a database service. In general, it is necessary to create a new server connection, connect to the server to list the available services, and add the service to the QGIS project.

With QGIS, the composition of maps is swift and the program is capable of printing or exporting to image and graphic files. The Print Composer presents a blank sheet of paper for the map crafting. Apart from the map body, the user could add images, text, legend, a scale bar, graphic shapes, arrows, attribute tables, and HTML frames. Map elements

become graphics on the composition canvas. The user could customize the properties of the map, the size of the paper etc. The Atlas generation tab allows the generation of a map book. For example, a municipality could generate an atlas by using a map sheet GIS layer and specifying which attribute column contains the map sheet number for each polygon. The Items tab allows toggling individual map elements on and off.

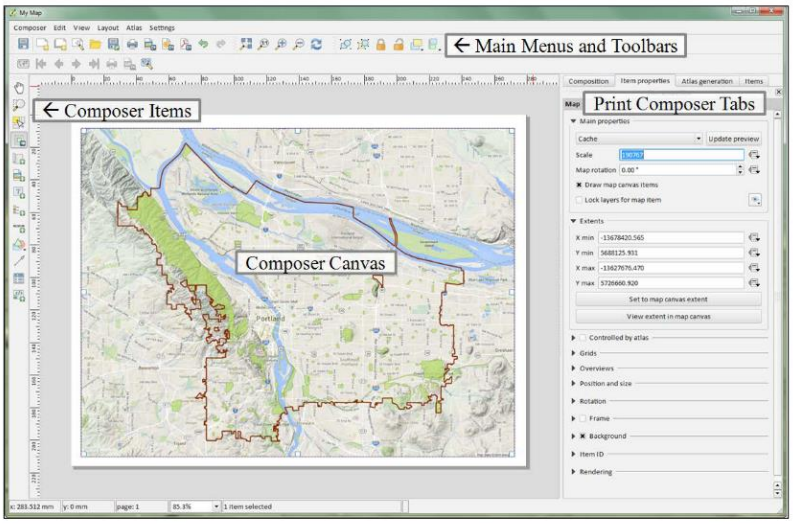


Figure 3.1: Composing maps in QGIS environment (QGIS manual).

3.1.3 Adding functionality with plugins

Even though the existence of potential workflows, analysis settings, and datasets within the broad field of GIS are numerous, no out-of-the-box software could contain the tools for every scenario. Fortunately, QGIS has been developed with plugin architecture from ground up. Plugins are add-ons to QGIS that provide additional functionality. Some are written by the core QGIS development team and others are written by the QGIS community. Some useful and well known plugins are being presented below.

OpenLayers Plugin

This plugin allows the user to insert an OpenStreetMap, Google, Bing map as a layer. Using the panel, the map type can be selected. The raster image that is been loaded could be used as a backdrop to help find out the location on the map. Figure 3.2 illustrates the application of this plugin. This plugin can be very useful in a hydrologic study as it allows the user to assess the real world terrain characteristics from real and recent satellite images.

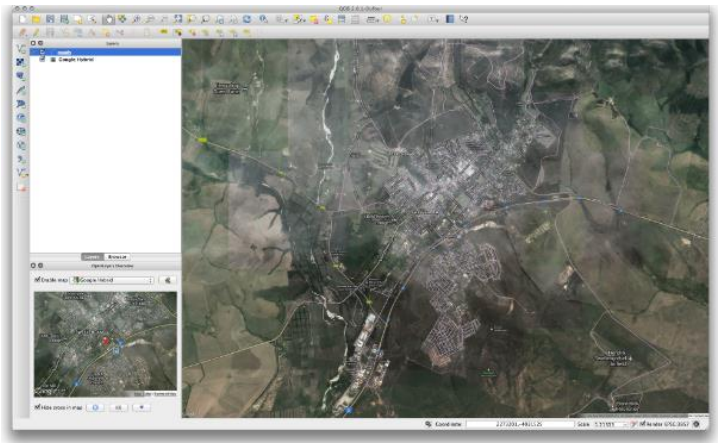


Figure 3.2: OpenLayers Plugin interface (OpenLayers manual).

Floodrisk plugin

This open source and free analysis toolbox, part of the open-source geographic information system Quantum GIS, was developed for estimating flood impacts due to flooding and, hence, to help authorities understand better and manage flood risk. These tools are not intended to be business-ready software applications; however, it is usable by third parties for evaluation and demonstration purposes.

The tool performs simple risk assessments, considering fixed event scenarios, estimates the probability of each scenario separate and calculates the consequences deterministically. In order to perform these tasks, the hazard, receptor and vulnerability are required.

In order to quantify the flood hazard, the maximum depth values and the maximum velocity values maps. In general, the inundation map is essential. Since timely flood warnings can save lives, warning time is a crucial data to assess the consequences for people. Therefore, the tool needs as input the information of the zones with different warning time (denoted as warning time map). The warning time indicates the amount of time between the reception of a warning and the instant in which the population of each structure could be affected by the flood event, i.e. the amount of time in which the population of each structure can mobilize or adopt mitigation measurements.

Receptors are considered the exposure, referring to people's assets and activities, threatened or potentially threatened by a hazard. The exposure data of the study area is stored in a geo – database. The dataset must consist of the following maps:

- the polygon boundary of the study area
- census map of population
- buildings and/or land use map
- lines maps of infrastructures (e.g. roads, railways etc.)

Receptors are defined in this case as characteristics of a system that describe its vulnerability. Catastrophic floods, such as those by dam-break or levee failure, can cause fatalities. In this case, the parameter "fatality rate" (or more presicely the percentage of the population at risk of death) is generally adopted to quantify vulnerability. Fatality rates are based on flood severity, warning time and warning quality.

There are empirical methodologies of literature calibrated on historical cases that provide values of fatality rates as a function of the parameters mentioned above. FloodRisk tool contains some of these tables of values. A flood can cause, also, many types of economic damages that can be classified in a variety of ways. The tool is able to calculate the tangible direct physical damages, i.e. the damages resulting from floodwaters on property and structures. Tangible damages are usually quantified and measured as monetary losses. Flood damages depend on many variables. These variables might include depth of water, velocity of floodwaters, duration of flood, sediment load, and contamination. However, flood damage to structure is strongly dependent on the water depth of a flood (Merz *et al.*, 2010).

Geographical information systems (GIS) tools are ideal to manage spatial information, providing adequate spatial processing and visualization of results. The tool, takes into account the quality and dissemination nowadays reached by Geographic Free/Open-Source Software (GFOSS) and has it in order to make available the results of the project. The main advantage of QGIS relies on the easiness and quickness in developing new plug-ins, using Python language. Therefore, this project was developed in QGIS platform and the interface was created in Python. FloodRisk toolbar is shown in the following Figure. Each button is linked to a window with

several options, such as menus, labels, edition windows, combo boxes, and simple buttons, that help the user to access input and output directories.

Example of loss maps produced using FloodRisk engine and presenting the expected economic losses for a benchmarking study case, proposed by the organizer committee of the 12th ICOLD International Benchmark Workshop on Numerical Analysis of Dams, are presented in Figure 3.4.

In conclusion the tool can be used to prioritize corrective actions to achieve an informed risk reduction or for the identification of the "optimal" measures of risk mitigation.

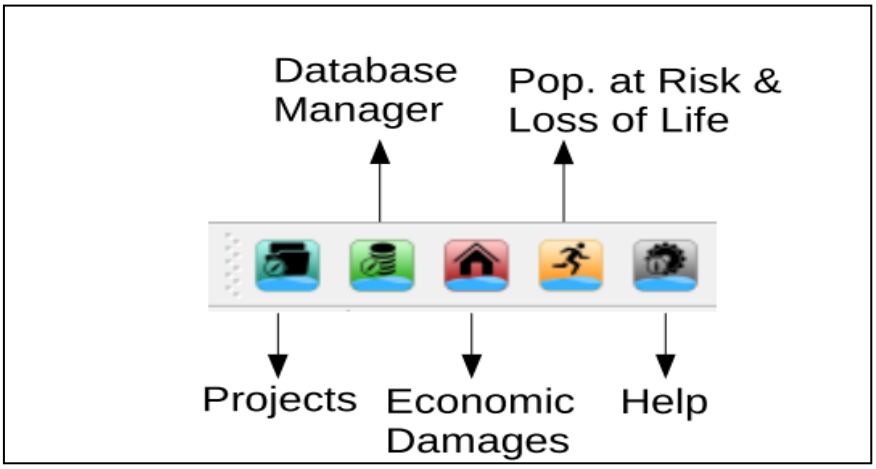


Figure 3.3: FloodRisk toolbar (FloodRisk manual).

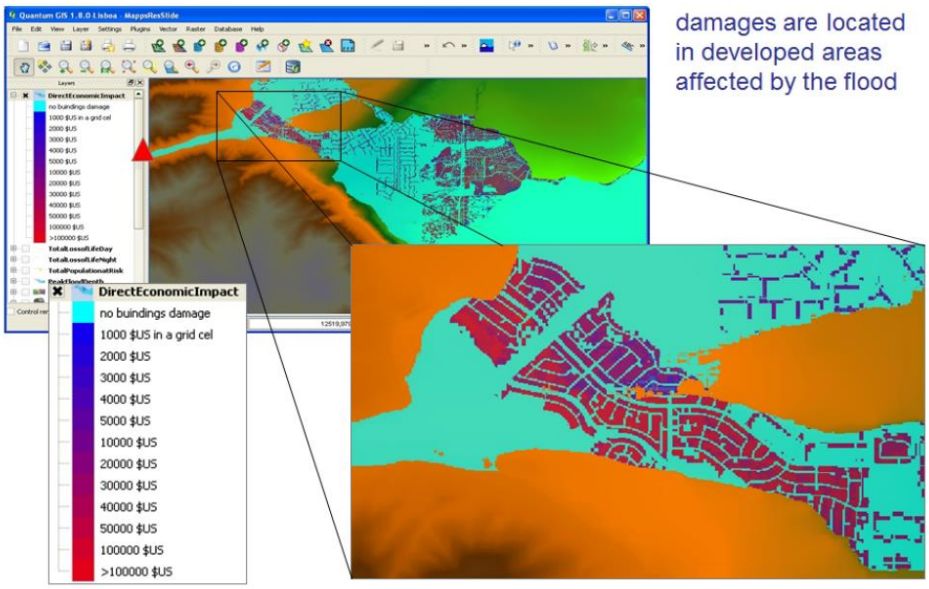


Figure 3.4: Example of damages map (FloodRisk manual).

RiverGIS

RiverGIS is a QGIS plugin for creating HEC – RAS flow model geometry from spatial data. The functionality is similar to that of HEC – GeoRAS. For data store and spatial operations it needs a PostGIS database. RiverGIS is free software and is released under the GNU General Public License.

RiverGIS is capable of creating a HEC – RAS model (1D and 2D). The fundamental difference from HEC – GeoRAS is that the RiverGIS uses a PostgresSQL database with the PostGIS spatial extension for data storage. A single PostgreSQL database can be used to store many

model geometries. Each model goes to its own schema, a kind of database directory for data grouping. Therefore, the first step is to create a new schema for a model. HEC – RAS 1D flow model geometry consists of rivers network, cross-sections and, optionally, hydraulic structures such as weirs, bridges or storage areas. Users have an option to import spatial data to the database from other data formats (i.e. ESRI Shapefiles) or create it from scratch.

The second step is the definition of the geometry data, its creation or import. These data are stored in a river database table, containing river lines, cross-sections etc. If a table needs a user's specified attribute, it is given in the user defined attributes column. Attribute names of the source data can differ from the database attribute names, but can be mapped easily to the right column, as shown above. If the required attributes are empty or nonexistent, users have to fill the database columns by hand after the import.

After that definition of the river network, the cross sections need to be presented. The elevation tool generates points along the cross-sections. RiverGIS also has the possibility of representing hydraulic structures, e.g. bridges, culverts etc.

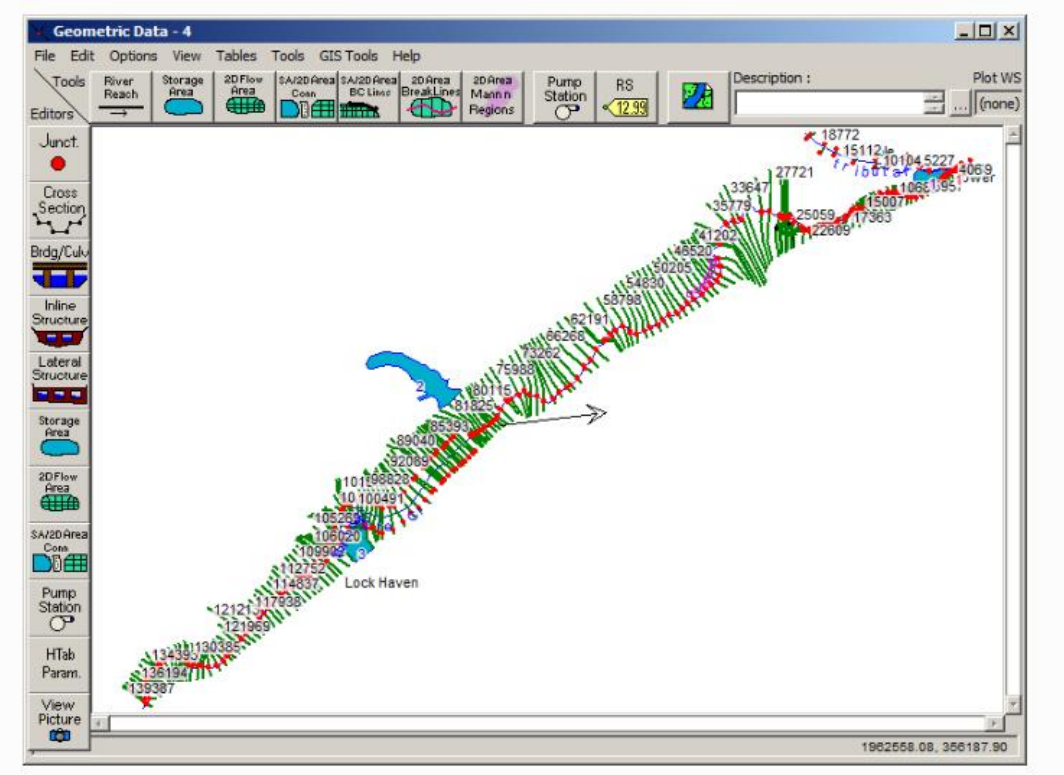


Figure 3.5: RiverGIS interface (RiverGIS manual).

As Figure 3.5 illustrates, the plugin can generate a model to import into HEC-RAS. RiverGIS builds 2D HEC – RAS geometry using the following river database tables created by a user:

- FlowAreas2D: a polygon layer representing 2D Flow Areas. It has 2 user defined attributes the 2D Flow Area name and the mesh cell size for a flow area.
- BreakLines2D: a polyline layer for aligning cell faces along the breaklines with 3 user defined attributes, the default mesh points spacing along a structure, the default mesh points spacing across a structure and the number of mesh rows that should be aligned to a breakline.

- BreakPoints2D: a point layer for creating a cell face at exact locations along the breaklines (optional). No attributes required.
- DTM: a digital terrain raster layers set.

The creation of the 1D base model with available 1D tools is considered to be the base model for the 2D flow areas. By defining the attributes of the FlowAreas2D and BreakLines2D, the 2D mesh is created (Figure 3.5).

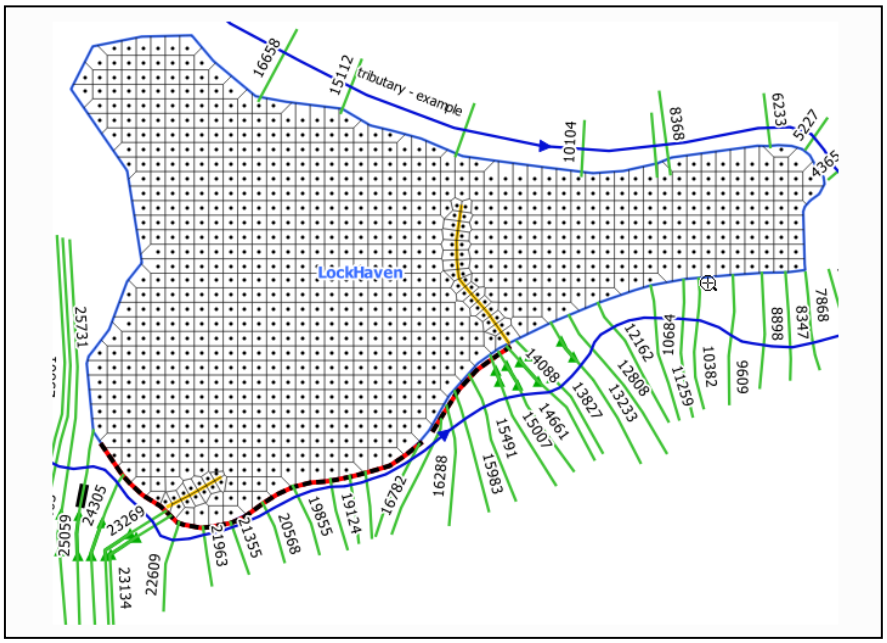


Figure 3.5: 1D model in RiverGIS (RiverGIS manual).

FreeWAT

FreeWAT platform is a large plugin integrated into QGIS. FREEWAT includes several modules for dealing with water management issues, with particular attention to groundwater. Simulation codes (mainly from the MODFLOW USGS family) for dealing with groundwater-related processes (e.g., groundwater flow, solute transport in aquifers, etc.) constitute the basis of the plugin. he complete list of modules so far integrated is provided below:

- Observations Analysis Tools (OAT) for time series analysis.
- Tools for analysis, interpretation and visualization of hydrogeological data (akvaGIS)
- Tools for analysis of groundwater quality datasets (akvaGIS)
- Groundwater flow modelling (based on MODFLOW-2005)
- Solute transport in the unsaturated zone (based on MT3D-USGS and the USB module)
- Solute transport in the saturated zone (based on MT3DMS)
- Density-dependent groundwater flow (based on SEAWAT)
- Management of water in agriculture (based on the FARM Process)
- Water management and planning (based on MODFLOW-OWHM)
- Crop yield at harvest (based on the Crop Growth Module, from the EPIC family)
- Sensitivity analysis and calibration (based on UCODE_2014)

The Figure 3.6 shows how the different modules are interconnected, taking as reference a standard modelling procedure.

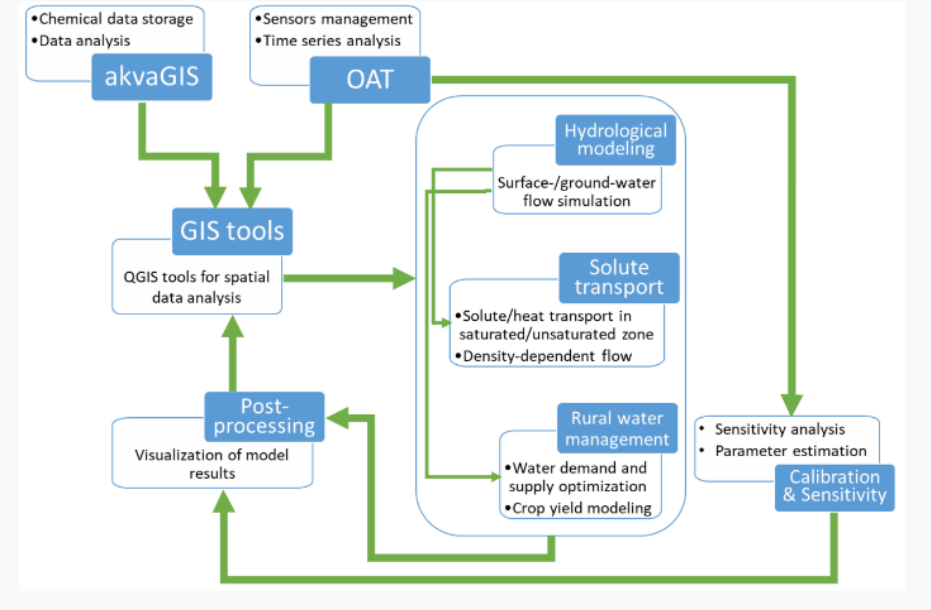


Figure 3.6: Flow – chart of the different modules in FreeWAT (FreeWAT manual).

3.2 GRASS

3.2.1 General Information

Geographic Resources Analysis Support System (commonly referred to as GRASS GIS), is a Geographic Information System (GIS) used for data management, image processing, graphics production, spatial modelling, and visualization of many types of data. It is an Open Source Software released under General Public License (GNU). GRASS GIS is an official project of the Open Source Geospatial Foundation.

Originally developed by the U.S. Army Construction Engineering Research Laboratories (USA-CERL, 1982-1995), a branch of the US Army Corp of Engineers, as a tool for land management and environmental planning by the military, GRASS GIS has evolved into a powerful utility with a wide range of applications in many different areas of applications and scientific research. GRASS is currently used in academic and commercial settings around the world, as well as many governmental agencies including NASA, NOAA, USDA, DLR, CSIRO, the National Park Service, the U.S. Census Bureau, USGS, and many environmental consulting companies.

In September 2006, the GRASS Project Steering Committee was formed which is responsible for the overall management of the project. The PSC is especially responsible for granting SVN write access.

GRASS GIS contains over 350 modules to render maps and images on monitor and paper, manipulate raster and vector data including vector networks, multispectral image data processing and create, manage, and store spatial data. GRASS GIS offers both an intuitive graphical user interface as well as command line syntax for ease of operations.

3.2.2 Capabilities

These are some of main capabilities that the program has to offer:

- **Raster analysis**: Automatic rasterline and area to vector conversion, Buffering of line structures, Cell and profile dataquery, Colortable modifications, Conversion to vector and point data format, Correlation/covariance analysis, Expert system analysis, Map

algebra (map calculator), Interpolation for missing values, Neighborhood matrix analysis, Raster overlay with or without weight, Reclassification of cell labels, Resampling (resolution), Rescaling of cell values, Statistical cell analysis, Surface generation from vector lines.

- **3D-Raster analysis**: 3D data import and export, 3D masks, 3D map algebra, 3D interpolation (IDW, Regularised Splines with Tension), 3D Visualization (isosurfaces), Interface to Paraview and POVray visualization tools.
- **Vector analysis**: Contour generation from raster surfaces (IDW, Splines algorithm), Conversion to raster and point data format, Digitizing (scanned raster image) with mouse, Reclassification of vector labels, Superpositioning of vector layers.
- **Point data analysis**: Delaunay triangulation, Surface interpolation from spot heights, Thiessen polygons, Topographic analysis (curvature, slope, aspect), LiDAR
- **Image processing**: Support for aerial and UAV images, satellite data (optical, radar, thermal), Canonical component analysis (CCA), Color composite generation, Edge detection, Frequency filtering (Fourier, convolution matrices), Fourier and inverse Fourier transformation, Histogram stretching, IHS transformation to RGB, Image rectification (affine and polynomial transformations on raster and vector targets), Ortho photo rectification, Principal component analysis (PCA), Radiometric corrections (Fourier), Resampling, Resolution enhancement (with RGB/IHS), RGB to IHS transformation, Texture oriented classification (sequential maximum a posteriori classification), Shape detection, Supervised classification (training areas, maximum likelihood classification), Unsupervised classification (minimum distance clustering, maximum likelihood classification).
- **DTM-Analysis**: Contour generation, Cost/path analysis, Slope/aspect analysis, Surface generation from spot heights or contours.
- **Geocoding**: Geocoding of raster and vector maps including (LiDAR) point clouds.
- **Visualization**: 3D surfaces with 3D query (NVIZ), Color assignments, Histogram presentation, Map overlay, Point data maps, Raster maps, Vector maps, Zoom/unzoom – function.
- **Map creation**: Image maps, Postscript maps, HTML maps.
- **SQL-support**: Database interfaces (DBF, SQLite, PostgreSQL, mySQL, ODBC).
- **Geostatistics**: Interface to "R" (a statistical analysis environment), MATLAB, etc.
- **Temporal framework**: support for time series analysis to manage, process and analyze (big) spatio-temporal environmental data. It supports querying, map calculation, aggregation, statistics and gap filling for raster, vector and raster3D data. A temporal topology builder is available to build spatio-temporal topology connections between map objects for 1D, 3D and 4D extents.
- **Furthermore**: Erosion modelling, Landscape structure analysis, Solution transport, Watershed analysis.

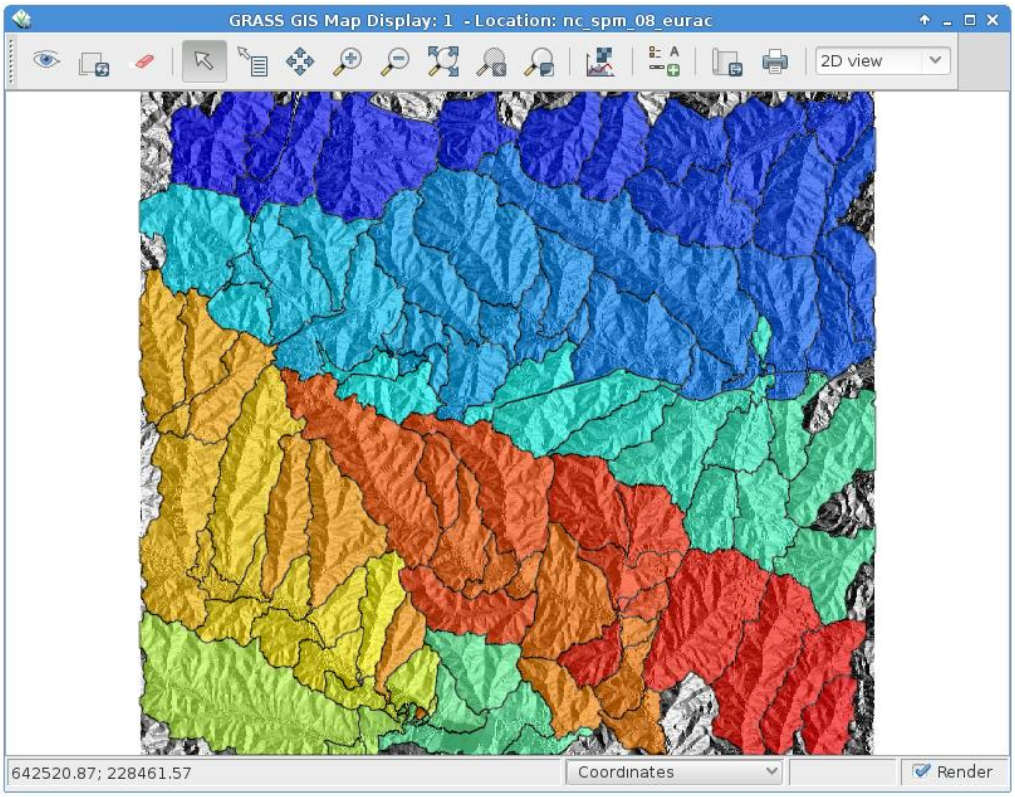


Figure 3.7: Vector data analysis in GRASS GIS (GRASS GIS manual).

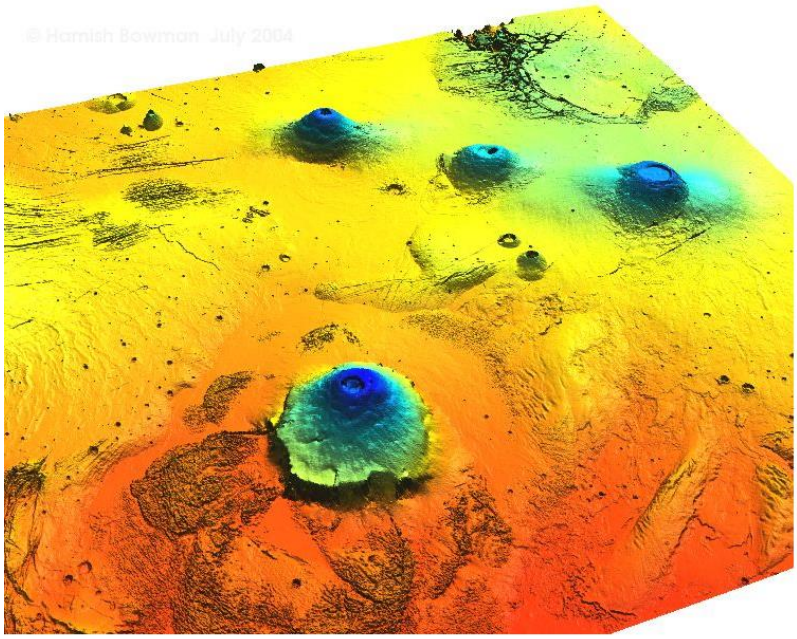


Figure 3.8: Mars topography from Mars Global Surveyor MOLA data (GRASS GIS manual).

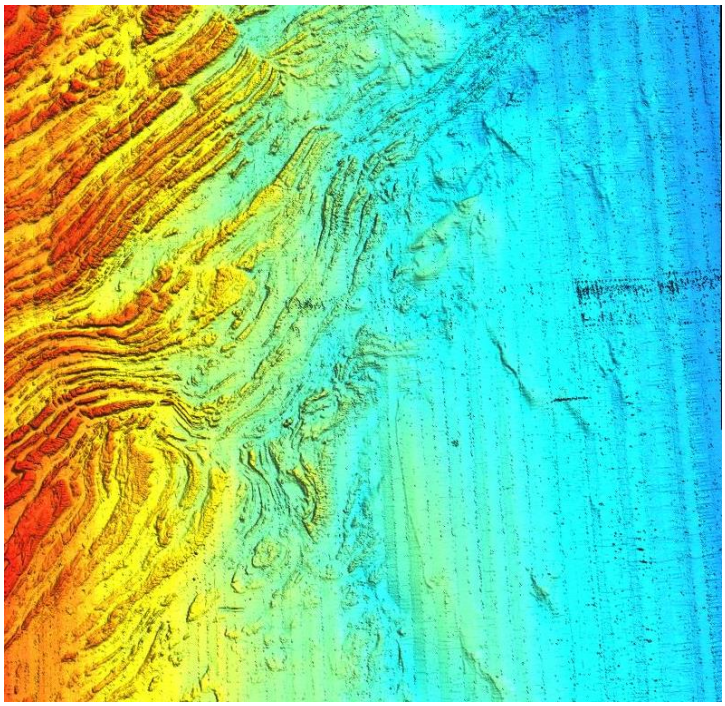


Figure 3.9: Floor bathymetry from MB – System (GRASS GIS manual).

As far as hydrological sciences are concerned, GRASS GIS offers a variety of possibilities for the hydrologists. The most significant ones are listed below:

- Itzï: A dynamic, fully distributed hydrologic and hydraulic model that simulates 2D surface flows on a regular raster grid using simplified shallow water equations. It uses GRASS GIS as a back-end for reading entry data and writing results.
- r.topmodel: Simulates TOPMODEL which is a physically based hydrological model (see section 2.3.4).
- HydroFOSS: A distributed, physically based hydrological model.
- SWAT: A river basin scale model developed to quantify the impact of land management practices in large, complex watersheds (see section 2.3.5).
- r.water.fea: Interactive program that allows the user to simulate storm water runoff analysis using the finite element numerical technique. Infiltration is calculated using the Green-Ampt formula. r.water.fea computes and draws hydrographs as well as at stream junctions in an analysis area. It also draws animation maps at the basin level.
- GIPE: The GRASS Image Processing Environment (GIPE) employs USLE, energy-balance and radiance-reflectance correction models (r.hydro.CASC2D; a physically-based, distributed, raster hydrological model which simulates the hydrological response of a watershed subject to a given rainfall field.
- r.gwflow: Numerical calculation program for transient, confined and unconfined groundwater flow in two dimensions.
- r3.gwflow: Numerical calculation program for transient, confined groundwater flow in three dimensions.
- r.sim.sediment: Sediment transport and erosion/deposition simulation using path sampling method (SIMWE).
- r.stream.basins: Delineates basins according to the stream network.
- r.stream.channel: Calculates local parameters for individual streams.
- r.stream.distance: Calculates distance to and elevation above streams and outlet.

- r.stream.order: Calculates Strahler's and more streams hierarchy.
- r.stream.segment: Divides network into near straight-line segments and calculate its order.
- r.stream.slope: Calculates local parameters for slope subsystem.
- r.stream.snap: Snap point to modelled stream network.
- r.stream.stats: Calculates Horton's statistics for Strahler and Horton ordered networks created with r.stream.order.
- r.stream.angle: Route azimuth, direction and relation to streams of higher order.
- r.stream.basins: Calculate basins according user input.
- r.stream.del: Calculates downslope length of first order streams and delete them if it length (in pixels) is lower than the threshold.
- r.stream.distance: Calculate distance to and elevation above streams and outlets according user input. It can work in stream mode where target are streams and outlets mode where targets are outlets.
- r.stream.extract: Stream network extraction. It produces a vector network with the direction of the vector lines corresponding to the flow direction.
- r.stream.order: Calculate Strahler's and Horton's stream order Hack's main streams and Shreeve's stream magnitude. It uses r.watershed or r.stream.extract output files: stream, direction and optionally accumulation. Output data can be either from r.watershed or r.stream.extract but not from both together.
- r.stream.pos: Route azimuth, direction and relation to streams of higher order.
- r.stream.stats: Calculate Horton's and optionally Hack's statistics according to user input.
- r.basins.fill: Generates a raster map layer showing watershed subbasins.
- r.water.outlet: Generates a watershed basin from a drainage direction map (from r.watershed) and a set of coordinates representing the outlet point of watershed.
- r.watershed: Watershed basin analysis program.
- r.lake: Fills a lake to a target water level from a given start point.
- r.basin: Generates the main morphometric parameters of the basin.
- r.threshold: Finds a first tentative value of upslope area to be used as input to extract the river network.
- r.hydrodem: Applies hydrological conditioning (sink removal) to a required input elevation map.
- r.sim.water: Overland flow hydrologic simulation using path sampling method (SIMWE).
- r.inund.fluv: Allows obtaining a fluvial potentially inundation map given a high-resolution DTM of the area surrounding the river and a water surface profile calculated through a 1-D hydrodynamic model.
- r.hazard.flood: Is an implementation of a fast procedure to detect flood prone areas. It may help in the delineation of flood prone areas, especially in basins with marked topography. The use of the modified topographic index should not be considered as an alternative to standard hydrological-hydraulic simulations for flood mapping, but may represent a tool for a preliminary delineation of flooding areas.

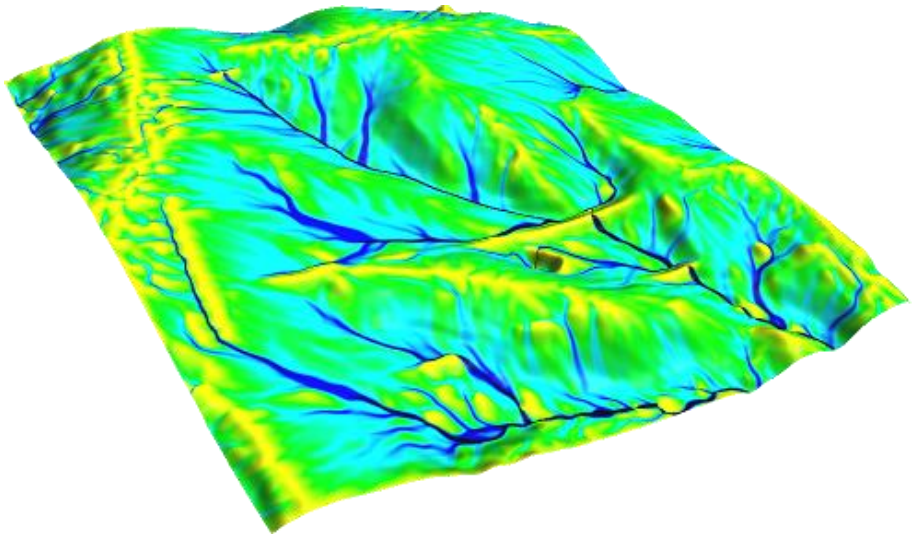


Figure 3.10: Example of flow accumulation map (GRASS GIS manual).

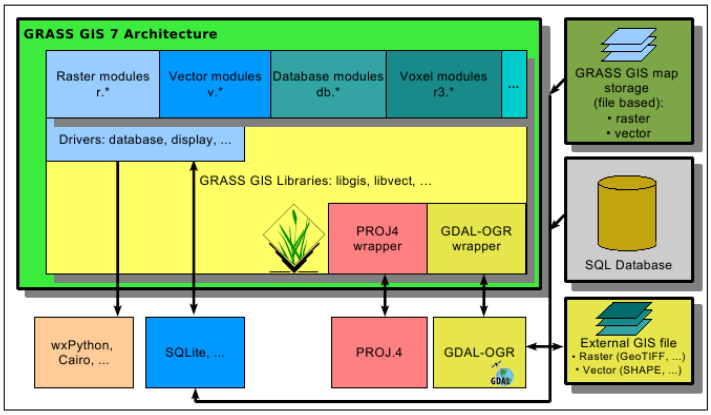


Figure 3.11: The GRASS 7 Architecture (GRASS GIS manual).

The architecture of the newest GRASS 7 is presented in Figure 3.11.

Being open – source, it is possible to view and change all of the source code, mostly C, with some bash and Python, in which it is written. GRASS is well-vetted and fully-functional. Being open-source means that you can view and change the source. GRASS is more of a computational / scripting GIS and less of a point-and- click GIS than ArcGIS. It is considered to be an advantage, however f needed there is also available an interactive surface.

Another major feature of GRASS is that it is a topological GIS, thus it is impossible to have small gaps or overlaps between vector areas. It also forces lines to meet and interact according to some fairly logical rules. This ensures consistency in geologic mapping and allows users to query vector maps based on their neighbors.

4 Why Python?

Python is the main computer language used in the software application developed within this thesis. This chapter provides some insights in the rationale behind the aforementioned choice and why the Python ecosystem is a mature environment for developing modern hydrological applications.

4.1 General characteristics and benefits of using Python

The design focus of the Python language is on productivity and code readability through an interactive python console, the clear, readable syntax (through the whitespace indentation as opposed to Java, C# etc. which use complex indentation), the full modularity, which supports hierarchical packages, and the dynamic data types and automatic memory management. What distinguishes Python from most of the other programming languages is its simplicity and powerfulness. This programming language can join together code snippets, which were written originally in C, C++, Fortran etc., and merge them with native Python code without hassle, thus allowing scientists from many fields to easily port their existing code, models and tools. There is a growing user community which makes many tools easily available as Python packages. It is worth mentioning that the Python Package Index, which is one major host of Python code, has more than 15,000 packages listed, indicating its current popularity. These packages include visualization, numerical algebra libraries, optimization toolboxes, geospatial libraries, interconnection with compiled and interpreted languages, memory catching, Web services, mobile and desktop graphical user interface programming, and many others.

Due to the access to a nice combination of GIS tools, mathematics, statistics etc., Python is a useful language for the science community. Also, it is OS-agnostic and scalable, making it compatible for users in every platform, from Linux supercomputers to Raspberry Pi units in custom remote measurement stations. In addition, its natural syntax, makes programs written in Python exceedingly clear and easy to read, especially for beginners, and is a great first programming language, especially for scientists with limited exposure to computer programming and students in an academic environment.

Python is a modern, interpreted, object-oriented, open-source and free language used in all kinds of software engineering in the last few years. This language is considered to be a robust integration platform, ranging from data analysis to distributed computing, and graphical user interfaces to geographical information systems (GIS). Due to the interpreted nature, Python allows an easy development and fast prototyping. While there are many other languages that have similar features, Python offers interconnectedness and comprehensiveness, allowing users to apply innovations from other communities and disciplines.

Every Python package is used in the same interpreted environment and also data flow does not have to occur through files. This makes it possible to access any variable in any tool at any (logical) time in the workflow. Python users have much greater ability to access innovations from industries outside of the Earth sciences. This latter advantage will become increasingly more important, as the science community enters the world of cloud computing, big data, and mobile computing. In the next decade, Earth sciences, like hydrology cannot afford to ignore these innovations.

The unique benefits of Python have begun to be recognized in the engineering community, and as a result its population of users is growing. In particular, Python can help the engineers to improve their research, through the clear code that offers. With the access to the new

capabilities being generated daily from industries, more and more engineers and scientists become aware of Python's abilities and its tools. It is considered to be the next wave in the computational Earth sciences.

4.2 Possible drawbacks and remedies

It is (generally) true that Python, as an interpreted language, is a relatively 'slow' language for computer simulations, as opposed to compiled languages, such as C++. However, the execution time alone is not the major factor that contributes to the "time cost" of a computational project. The costs of prototyping, development and maintenance work are also significant. However, these procedures are inherently done with lower time cost in Python. Nowadays, for research programming, it may be far more economic to accept that the code runs only at 25% of the expected possible speed if this saves for example a month of a researcher's time.

Furthermore, research code is usually not limited to one project, but has long run life expectancy. By using it repeatedly, the code evolves, grows, bifurcates etc. It is essential that the programmer invests the time to make the code fast and after works on speed optimization. So, the use a language that is easy to read and has a great expressive power will help in this direction. Additionally, a well-written Python code could be very fast if time critical parts in executed through compiled language or JIT-interpreters (Just In Time compilation – see the section about the numba package). As long as these calculations are done efficiently, the overall time will be insignificant for most computational projects.

4.3 Comparison to other languages commonly used in hydrology

4.3.1 Python vs MATLAB

The most common implementation of Python is in C (also known as CPython), which is mostly referred to as "Python". Apart from the programming language and interpreter, Python also consists of an extensive standard library. This library is aimed at programming in general and contains modules for OS-specific stuff, threading, networking, databases, etc.

On the other hand, MATLAB is a commercial numerical computing environment and programming language. The concept of MATLAB refers to the whole package, including the IDE. The standard library does not contain as much generic programming functionality but does include matrix algebra and an extensive library for data processing and plotting. For extra functionality the Mathworks provides toolkits.

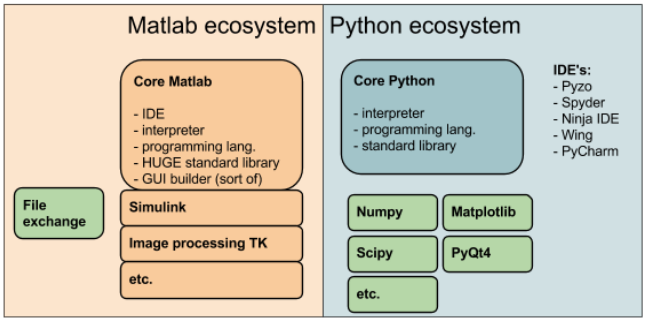


Figure 4.1: Matlab vs Python.

As Python is open and free, it is very easy for other parties to design packages or other software tools that extend Python. It is possible to create applications using any of the mayor GUI libraries (e.g. Qt), use OpenGL, drive your USB port, etc. Another example is the Cython

package that enhances the speed of algorithms by converting Python to compiled C code, and cx_Freeze that creates standalone application from source code.

Each package is being developed by a different group (though there are common overlaps), who are also users of the package. Many packages are available for different purposes. In this open source ecosystem most packages are driven by a handful of core developers, but many of a package users contribute to the development by reporting issues, helping with documentation, and making small improvements to the code, debugging, issue handling, etc.

MATLAB has some fundamental shortcomings, most of which arise from its commercial nature:

- The algorithms are proprietary, which means the code of most of the algorithms is not available to the users, leaving them with the doubt that the code was not implemented correctly.
- MATLAB is quite expensive, which means that code that is written in MATLAB can only be used by people with sufficient funds to buy a license.
- Usually, Mathworks puts restrictions on code portability, the ability to run code on someone else's computer.
- The proprietary nature also makes it difficult almost impossible for third parties to extend the functionality of MATLAB.

Furthermore, there are some other issues that stem from MATLAB's origins as a matrix manipulation package:

- The semicolon, which can be useful to show the result when you type code in the console, in scripts is partly redundant.
- Indexing is done with braces rather than brackets, making it difficult to distinguish it from a function call.

However, MATLAB has major advantages, for example:

- It has a solid amount of functions.
- Simulink is a product for which there is no good alternative yet.
- It is easier for beginners, because the package includes everything someone would need, while in Python the installation of extra packages and an IDE is essential.
- It has a large scientific community, meaning it is used on many universities (although few scientists on their own invest in licenses, nor do small companies).

Generally, Python advantages out-weight MATLAB's. Some of them are listed below:

- Open source program
- Python was created to be a generic language that is easy to read, while MATLAB started as a matrix manipulation package, and then followed the introduction of a high level programming language.
- Powerful, meaning it is a language well designed, with powerful datatypes such as lists, sets and dictionaries, helping the programmer to organize the data.
- MATLAB supports namespaces for the functions that the programmer writes, but the core of MATLAB is without namespaces. Python works with modules, which is needed to be imported. Using namespaces gives structure to a program. In Python everything is an object, so each object has a namespace itself, making this language good at introspection.

- Introspection allowing the user to access any part of the application, including some of Python's internals.
- String manipulation is deeper integrated within Python.
- Portability, making possible to program in most of the operating systems (Windows, Linux, and OS X).
- Functions and classes can be defined anywhere. For example, one file (whether it is a module or a script) could contain many functions and classes.
- Python allows the creation of GUI (Graphical User Interface) in order to transform a program to an application which is presentable and functional as well. There are major GUI toolkits like Wx or Qt. MATLAB's GUIDE approach is lackluster compared to these toolkits.

Conclusively, Python is considered to be more appropriate for creating a standalone program that MATLAB.

4.3.2 Python vs C/C++

The C programming language is a low-level compiled language (sometimes classified as a 3rd generation language) that is widely used in academia, industry and commerce. C++ provides a different programming paradigm than C but for the purpose of this work, C++ is more similar to C than it is to MATLAB or Python. The main advantage of compiled low-level languages is their execution speed and efficiency (for example in embedded systems). C/C++ is able to create more compact and faster runtime code, making it the language of choice for 95% of operating systems' code. When it comes to speed, however, runtime speed is not the only aspect of development to consider, it is crucial to think about the development speed. While Python may be less efficient than C/C++ at runtime, during development it's much more efficient. Interpreters read each line of code, parse it, do runtime checks and call routines in order to execute the operations in the code. This is a lot more activity than what you get from running C/C++ code, where the same line of code might be compiled into just a couple of instructions. This can lead to slower runtime speeds and higher energy consumption with Python.

C++, which originally designed to move C programmers to a higher level, aimed for functionality more than error prevention. As a program language is complicated and difficult to learn. One the other hand, one of the main advantages of this language is that the user can achieve almost everything from writing a new OS to shader applications for modern Graphics-heavy video games. However, there are some serious problems with libraries, due to its aging design. Its specification and diagnostics are often baffling. Its standard library is large, but unfortunately not powerful. Furthermore, some of the commonly used class libraries, e.g., Boost, CERN ROOT, CGAL etc., are often complicated and idiosyncratic.

Regardless the fact that C++ is complicated, there are quite few benefits. Firstly, the user is able to create its own data structures. Secondly, if the programmer needs high efficiency, this is the language of their choice. Furthermore, if someone needs assembler level coding, C++ is a better option than Python, but these benefits do not matter match in the context of programming for hydrology applications.

Contrary to C/C++, Python is high-level language, and it is clear that for engineers, who ask for an easy to implement programming language to solve their problems, Python seems to be a better choice.

4.3.3 Python vs Fortran

Fortran falls into the same category with C, but while Fortran is still commonly used in academia it appears to be overtaken by C (and C++) in many industrial applications.

FORTRAN, is approximately the 1/3 size of C++ and a much simpler language. FORTRAN, which is still traditionally used for scientific/engineering numerical computing for reasons of inertia, t has a very long history of compiler optimization work. It was designed before there was real language/compiler science. It is a compiled imperative programming language, originally developed by IBM in the 1950s for scientific and engineering applications. FORTRAN came to dominate this area of programming early on and has been in continuous use for over half a century in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing and is used for programs that benchmark and rank the world's fastest supercomputers. In addition, it is a much more optimizable language than C++.

However, due in large part to disagreements among members of the Fortran development groups, Fortran 77 was deficient in a number of areas. Most or all of these have since been addressed by Fortran 90/95, however, so they do not represent current language deficiencies. These limitations include:

- Poor string handling, including weak concatenation and length functions.
- Subroutines pass arguments by reference rather than by value, making data protection difficult.
- Data scoping is limited. Variables can either be local or in COMMON blocks, but no other scoping is allowed. As a result, it's not possible to write file-level procedures; shared logic must be in a separate subroutine or repeated via cloning.
- Loop controls are somewhat limited, requiring continued use of the GOTO statement to manage flow in some cases.

4.3.4 Python for hydrologists

As mentioned before, Python is an easy enough language for everyone to use. Most engineers do not have the urge to be programmers. What they try to achieve is to use tools in order to make their work more efficient and faster than before. Hydrologists slowly make Python the programming language of their choice for all the above reasons and for the numerous libraries. Python is especially useful as glue for existing programs, either written in C or FORTRAN. Some of these are listed below:

- **GRASS GIS**, which has been interfaced with Python.
- **CFM**, a library for the creation of hydrological models.
- **MODFLOW**, the groundwater model is interfaced by FloPy.
- **OpenHydrology**, a library of open source hydrological software written in Python to operate as packages under an umbrella interface.
- **PyQGIS**, a Python interface to QGIS.

There are also many hydrologic applications entirely written in Python, such as:

- **AMBHAS**, a hydrological library in Python.
- **ANUGA 2**, a package for modelling dam breaks, riverine flooding, storm-surge or tsunamis (in Python and C).
- **Evaplib**, a Python library containing functions for calculation of evaporation rates. Functions include Penman open water evaporation, Makkink reference evaporation, Priestley-Taylor evaporation, Penman-Monteith evaporation and FAO's Penman-Monteith ET0 reference evaporation for short, well-watered grass. In addition, there is a function to calculate the sensible heat flux from temperature measurements.

- **EcoHydrolib**, which provides a series of Python scripts for performing ecohydrology data preparation workflows.

As far as hydrological models are concerned, these are some of the most commonly used:

- **EXP-HYDRO** Model, which is a catchment-scale hydrological model that operates at a daily time step.
- **LHMP**, a lumped hydrological model- tiny docker container with complete environment for predictions.
- **PyCatch**, a component based hydrological model of catchments built within the PCRaster Python framework.
- **Topoflow**, a Python hydrologic model by Scott Peckham.
- **Wflow**, which is a distributed hydrological model platform that currently includes two models: the wflow_sbm model (derived from the topog_sbm soil concept) and the wflow_hbv model which is a distributed version of the HBV model. This is actually part of a larger Deltares project called OpenStream.

GIS capabilities are also present:

- **pyDEM**, a Python digital elevation model analysis package. PyDEM depends on TauDEM for certain steps (e.g., pitfilling) and it also makes extensive use of the GDAL library for working with geospatial rasters.
- **PyGeoprocessing**, is a Python/Cython library that provides a set of commonly used raster, vector, and hydrological operations for GIS processing.

Tools for the field of Meteorology are also available in Python:

- **Meteolib**, is a Python library containing meteorological functions for calculation of atmospheric vapor pressures, air density, latent heat of vaporization, heat capacity at constant pressure, psychrometric constant, day length, extraterrestrial radiation input, potential temperature and wind vector. Functions to convert event-based data records to equidistant time-spaced records (event2time) and to convert date values to day-of-year values (date2doy) are now in a separate meteo_util module.
- **MetPy**, is an Open Source Python Toolkit for Meteorology.
- **Melodist** (MEteoroLOgical observation time series DISaggregation Tool), is an open source software package written in Python for temporally downscaling (disaggregating) daily meteorological time series to hourly. The model is documented by Forster *et al.* (2016).

Visualization is served through these packages/applications:

- **ggplot**, is a plotting system for Python based on R's ggplot2 and the Grammar of Graphics. It is built for making professional looking, plots quickly with minimal code.
- **VisTrails**, is an open-source scientific workflow and provenance management system that supports data exploration and visualization.
- **uvcmetrics** metrics, are diagnostics for comparing models with observations or each other. This is part of the Uv-CDAT website which contains also other visualization tools.

Tools for dealing with uncertainty and sensitivity analysis:

- Stijn Van Hoey tools

All above features make clear that Python is a suitable language for engineers, and specifically for Hydrologists, due to the numerous libraries and packages it has to offer.

4.4 Anaconda Distribution

Anaconda Distribution is preferred for the Python installation, as many scientific packages require a specific version of Python to run, making extremely difficult to keep them from interacting with each other, or to keep them updated. Anaconda is an open source, easy to install, high performance Python (and R) distribution, with the conda package/environment manager and a pre-installed collection of 1,000+ open source packages with free community support for interaction. Some of the packages are:

- NumPy: N-dimensional array for numerical computation (numpy.org)
- SciPy: Scientific computing library for Python (scipy.org)
- Matplotlib: 2D Plotting library for Python (matplotlib.org)
- Pandas: Powerful Python data structures and data analysis toolkit (pandas.pydata.org)
- Seaborn: Statistical graphics library for Python (seaborn.pydata.org)
- Bokeh: Interactive web visualization library (bokeh.pydata.org)
- Scikit – Learn: Python modules for machine learning and data mining (scikit-learn.org/stable)
- NLTK: Natural language toolkit (nltk.org)
- Jupyter Notebook: Web app that allows you to create and share documents that contain live code, equations, visualizations and explanatory text (jupyter.org)

After the installation of Anaconda, the Anaconda Navigator (Figure 4.2) is available. It is one easy way to use Python programs without having to use command line commands, which is essential for engineers who are not familiar with programming. In the Anaconda Navigator, the Spyder IDE is available (Figure 4.3). It is an interactive programming environment with several similarities to MATLAB. Spyder has an editor and an interactive shell. It also has an interface for debugging, inspectors for objects and documentation, and variable and folder explorers. This resembles MATLAB in useabilit. In the context of this thesis, Spyder is the IDE used to program in Python 3.6.

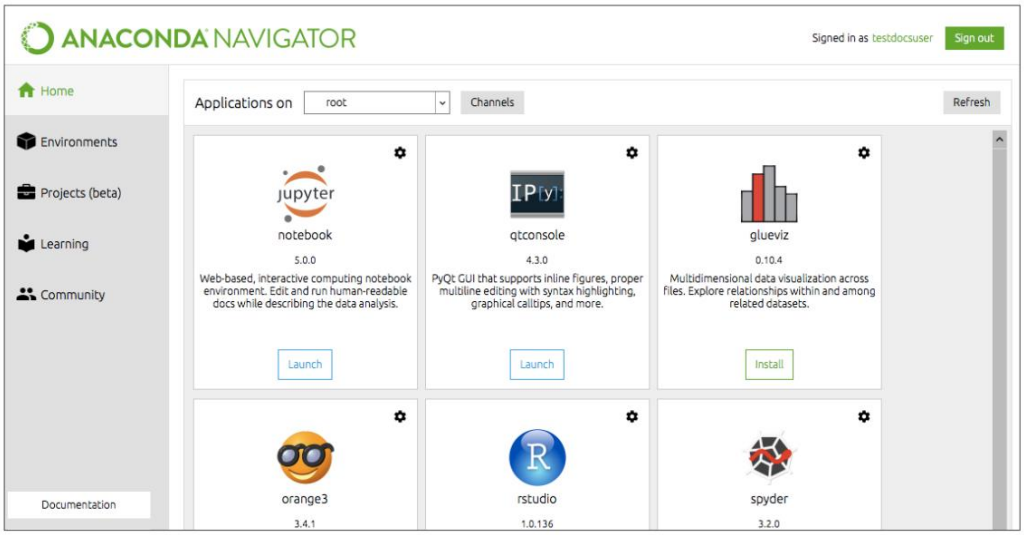


Figure 4.2: Anacoda Navigator environment.

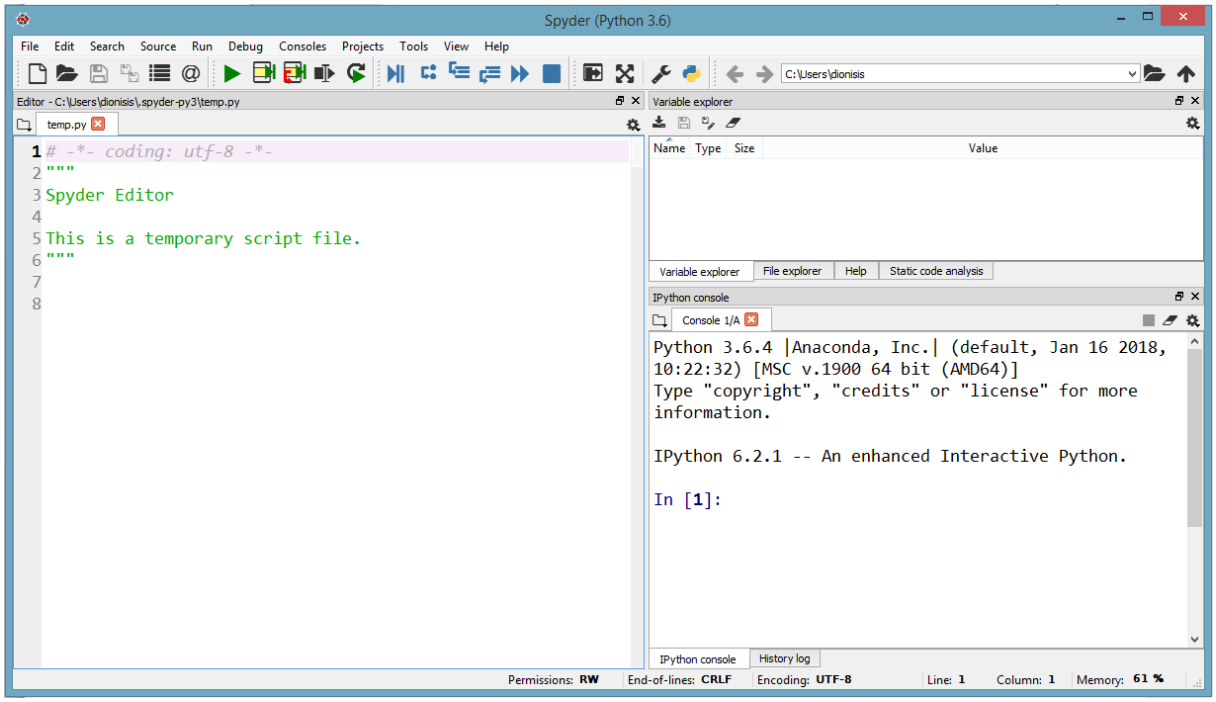


Figure 4.3: Spyder IDE.

4.5 Packages (Python 3.6) used in software development

4.5.1 Numpy

Numpy is the fundamental package for scientific computing with Python. It contains a powerful *n*-dimensional array object, broadcasting functions, tools for integrating C/C+ and Fortran code, as well as useful linear algebra, Fourier transform and random number capabilities. Numpy could be used as an efficient multi-dimensional container of generic data. The fact that arbitrary data types can be defined, allows NumPy to seamlessly and speedily integrate with a wide variety of databases. NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. Generally, in Python Numpy's array class, called ndarray, is an essential element when writing code. Creation of arrays, basic operations, indexing, slicing, iterating in one and two-dimensional arrays and linear algebra functions are some of the essential elements of this library.

4.5.2 GDAL (Geospatial Data Abstraction Library)

The GDAL project started by Frank Warmerdam in 1998, who worked as an independent professional on the GDAL/OGR library. GDAL is a translator library for raster and vector geospatial data formats that is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation. It presents a single raster abstract data model and single vector abstract data model to the calling application for all supported formats. It also comes with a variety of useful command line utilities for data translation and processing. The GDAL/OGR tree holds source for a vector IO library inspired by OpenGIS Simple Features. Although being separate from GDAL, currently both applications reside in the same source tree and they are somewhat entangled.

4.5.3 Geopandas

GeoPandas is an open source project for working with geospatial data in Python. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric

operations are performed by shapely. Geopandas further depends on fiona for file access and descartes and matplotlib for plotting.

The goal of GeoPandas is to work with geospatial data in Python easier. It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in python that would otherwise require a spatial database such as PostGIS.

GeoPandas implements two main data structures, a GeoSeries and a GeoDataFrame. These are subclasses of pandas Series and DataFrame, respectively. A GeoSeries is essentially a vector where each entry in the vector is a set of shapes corresponding to one observation. An entry may consist of only one shape (like a single polygon) or multiple shapes that are meant to be thought of as one observation.

Geopandas has three basic classes of geometric objects, i.e. points, lines and polygons. GeoDataFrame is a tabular data structure that contains a GeoSeries. The most important property of a GeoDataFrame is that it always has one GeoSeries column that holds a special status. This GeoSeries is referred to as the GeoDataFrame's "geometry". When a spatial method is applied to a GeoDataFrame (or a spatial attribute, like area, is called), this commands will always act on the "geometry" column. GeoPandas can read almost any vector – based spatial data format including ESRI shapefile, GeoJSON files etc. In conclusion, GeoPandas is able to handle multiple spatial datasets, performing various tasks.

4.5.4 Matplotlib

Matplotlib is a Python 2D plotting library originally developed by John Hunter, to produce publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib allows the user to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. Figure 4.4 illustrates an example.

For simple plotting the *pyplot* module provides a MATLAB-like interface, particularly when combined with IPython. The user has full control of line styles, font properties, axes properties, etc., via an object oriented interface or via a set of functions familiar to MATLAB users. Matplotlib ships with several add-on toolkits, including 3d plotting.

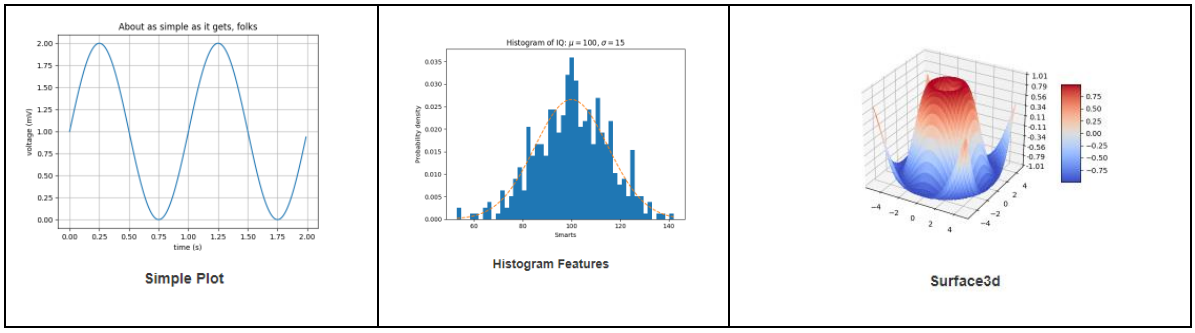


Figure 4.4: Matplotlib library's examples.

4.5.5 Numba

Numba is a compiler for Python array and numerical functions that gives you the power to speed up your applications with high performance functions written directly in Python.

Numba generates optimized machine code from pure Python code using the LLVM compiler infrastructure. With a few simple annotations, array-oriented and math-heavy Python code can be just-in-time optimized to performance similar as C, C++ and Fortran, without having to switch languages or Python interpreters.

Numba's main features are:

- on-the-fly code generation (at import time or runtime, at the user's preference)
- native code generation for the CPU and GPU hardware
- integration with the Python scientific software stack via the Numpy package
- multi-threading support

4.5.6 PyQt

PyQt is one of the most popular Python bindings for the Qt cross-platform C++ framework. PyQt developed by Riverbank Computing Limited. Qt itself is developed as part of the Qt Project. PyQt provides bindings for Qt 4 and Qt 5. This package allows the creation of rich GUI applications integrated with native Python code.

4.5.7 Pytictoc

Pytictoc contains a class TicToc that replicates the functionality of MATLAB's tic and toc for easily timing sections of code. Under the hood, pytictoc uses the default_timer function from Python's timeit module. It is essential for measuring performance of time-critical code, which is useful because most distributed hydrological models typically have larger execution time, due to their complexity.

4.5.8 Scipy library

The SciPy library is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. The optimization algorithm employed in this thesis for the calibration of the model is the

5 Modeling framework

5.1 Model overview

This chapter presents in detail the proposed event-based rainfall-runoff modeling procedure, employed within a distributed schematization of the river basin.

First, we distinguish the effective from the gross rainfall, at a cell basis, thus extracting the spatial distribution of surface runoff during the simulation period. The underlying model is based on an improved NRCS-CN scheme, which uses the so-called reference value of the CN value (different for each cell), and two lumped (i.e. common for the entire basin) dimensionless parameters, i.e. one for representing the antecedent soil moisture conditions (AMC) of the basin at the beginning of the storm event, and one for estimating the initial rainfall abstraction. The proposed scheme contains several novelties, regarding the estimation of the reference CN value (i.e. the value that refers to average soil moisture conditions and 20% abstraction ratio) and its adjustment against the two model parameters.

For the propagation of runoff to the basin outlet we consider two flow types, i.e. an overland flow across the catchment's terrain, and a channel flow along the river network. These two types are synthesized by employing a velocity-based approach, to form the flood hydrograph. This approach implements an original methodology for assigning realistic velocity values along the river network, also accounting for the novel concept of the varying (i.e. dependent on runoff intensity) time of concentration.

The proposed approach takes advantage of regional relationships and literature values for assigning appropriate values to all model attributes, expect for the two lumped parameters of the rainfall-runoff transformation, which are either manually assigned or inferred through calibration (provided that observed flow data are available). In the last case, it is essential to extract the baseflow from the total hydrograph, which may be done through several approaches of varying complexity. Here we propose an empirical method, requiring the fitting of a lumped hydrological model the observed hydrograph, which explicitly accounts for the contribution of baseflow to total runoff.

An alternative, more integrated approach, aims at running the distributed model with additional functionalities, in order to obtain the full hydrograph at the basin outlet. In this context, we have also developed a more generic version of the modeling framework, in which the NRCS-CN procedure is combined with a continuous soil moisture accounting scheme, thus generating both the surface (overland) runoff as well as the interflow through the unsaturated zone. Apparently, the augmented version of the model requires more parameters, since more processes are accounted for within the simulation procedure.

5.2 Model schematization and inputs

For the schematization of the model domain, the user needs to formulate two spatial layers, i.e. a grid-based partition of the basin to equally-dimensioned (squared) cells, and a graph-based configuration of the hydrographic network, comprising junctions and interconnected river segments. Both layers can be easily extracted on the basis of a digital elevation model (DEM) of the basin, using typical tools that are available in any GIS environment. The level of detail of the two spatial analyses is determined by the user, by assigning a proper cell size and a proper flow accumulation threshold.

Apart from the DEM, other geographical layers may be necessary within the estimation of spatially-distributed parameters of the modeling procedure, i.e.:

- Geology and/or soil permeability maps
- Land use/land cover maps
- Terrain slope maps (may be produced via the DEM)
- River sections and associated geometrical properties

Apparently, a mapping of rainfall data over a specific (typically short) time period is also essential input of the model. This map can be either directly obtained, from available distributed information (e.g. radar data) or, more often, extracted through interpolation of point rainfall observations at a number of stations in the broader area of the study basin.

Finally, in case of calibration, either the flood or the full hydrograph is required, depending on the model version to be used (surface model or full-process model).

5.3 Cell-based rainfall-runoff transformation via an improved NRCS-CN scheme

5.3.1 General modeling procedure

Within any event-based rainfall-runoff procedure of transforming a given rainfall event to flood runoff, it is essential to subtract the hydrological deficits, namely the part of rainfall that is initially intercepted in the ground and by the canopy, which next is either infiltrated or evaporated. The part of gross rainfall that transforms to surface runoff is known as *effective rainfall* or *rainfall excess*.

In order to extract the effective from the gross rainfall, one of the most widespread techniques is the SCS Curve Number approach, developed by the Soil Conservation Service (1972), which is currently referred to as Natural Resources Conservation Service Curve Number (NRCS - CN). This method has been globally used to model the rainfall-runoff processes, and has been included in several hydrological models, e.g. HEC-HMS.

The NRCS-CN approach is a simple empirically developed hydrological model that estimates the temporal evolution of surface runoff from a given rainfall event, based on the following assumptions (U.S. Department of the Interior, 1977; Koutsoyiannis and Xanthopoulos, 1999, p. 274-278):

- During an initial time interval, $t_{\alpha 0}$, the cumulative rainfall so far, h_{a0}, is transformed to deficit (herein referred to as initial deficit), without producing any runoff. Therefore, after time $t_{\alpha 0}$, the maximum effective rainfall depth, h_e, cannot exceed the potential quantity $h - h_{a0}$, where h is the total gross rainfall.
- The additional to h_{a0} deficit during a very large storm event cannot exceed a maximum quantity S, called *maximum potential retention*.
- At any time, $t > t_{\alpha 0}$, the ratios of the cumulative effective rainfall, h_e, and the total minus the initial deficit, $h_e - h_{a0}$, to the corresponding potential quantities $h - h_{a0}$ and S, respectively, are equal.

Under the above assumptions we get the empirical expression, in which all variables refer to cumulative quantities:

$$h_e = \begin{cases} 0 & h \le h_{a0} \\ \dfrac{(h - h_{a0})^2}{h - h_{a0} + S} & h > h_{a0} \end{cases} \quad \textbf{(5.1)}$$

The above formula is applicable not only for the total rainfall depth but also for any intermediate value, thus allowing obtaining the temporal evolution of surface runoff, h_e, and associated hydrological deficits, $h - h_e$, against time.

The model uses two parameters, i.e. the maximum potential retention, *S*, and the initial deficit, h_{a0}. Typically, SCS considers a linear relationship between the two quantities, i.e.:

$$h_{a0} = \lambda S \quad \textbf{(5.2)}$$

where λ is a dimensionless parameter, next referred to as *initial abstraction ratio*. In an attempt to avoid the need for calibration, NRCS suggests employing a standard value of 20%, which has been derived on the basis of field experiments, mainly implemented in small agricultural catchments with mild slopes. Under this premise, the governing equation (5.1) contains only one unknown, i.e. the maximum potential retention, *S*.

In our approach, we consider λ as a free parameter of the model, which is uniformly distributed over the basin (thus a common value is applied to all cells). On the other hand, the maximum potential retention, *S*, is handled as a distributed (i.e., cell-based) property (not parameter), depending on the spatially-varying physiographic characteristics of the study area. For its estimation we employ an original approach, which accounts for the so-called reference curve number value of each cell, the lumped parameter λ, and the antecedent soil moisture conditions at the beginning of the storm event, which are also expressed through a lumped dimensionless metric that describes the initial conditions of the model. In the following sections, we first describe the standard approach by NRCS for extracting S, and next explain the revised methodology, which is implemented in our model.

5.3.2 Standard NRCS-CN approach for estimating maximum potential retention

According to the classical practice by SCS, the maximum potential retention, *S*, is mapped into a dimensionless quantity, referred to as *curve number*, CN, via the well-known formula:

$$S = 254\left(\frac{100}{CN} - 1\right) \quad \textbf{(5.3)}$$

SCS has introduced this conceptual quantity in an attempt to capture the physiographic characteristics that affect runoff generation in a single value, ranging from 1 to 100 (the larger is this value, the larger is the runoff produced from a given rainfall event). According to SCS's standards, CN depends on soil and land characteristics, as well as on the soil moisture present in the soil profile before the start of a rainfall event. In this respect, it considers three antecedent soil moisture (AMC) conditions (type I: dry, type II: moderate, type III: wet), depending on the cumulative 5-day antecedent rainfall and the season (dormant or growing).

CN values for AMC II conditions and the typically-used ratio of initial abstraction losses, i.e. 20% of maximum potential retention (henceforth referred to as *reference conditions*) are determined from detailed lookup tables by NRCS (2004), accounting for several combinations of land use/land cover characteristics and four hydrological soil types (A, B, C, D). The latter are based on infiltration and transpiration rates, and they are further classified according to their hydrological conditions (good, fair, poor). These reference CN values have been extracted

experimentally, from rainfall and runoff measurements over a wide range of geographic, soil, and land management conditions.

Table 5.1: CN ranges across rural areas for AMC II conditions (adapted by Koutsoyiannis, 2011, p. 126).

Land cover	Hydrologic soil group			
	A	B	C	D
Cultivated areas	62-72	71-81	78-88	81-91
Pasture areas	30-68	58-79	71-86	78-89
Forests	25-45	55-66	70-77	77-83

In particular, soils falling in group A, B, C and D exhibit high, moderate, low, and very low rates of infiltration, respectively, thus CN increases as the soil type changes from A to D. The classification is made as follows:

- Group A: Typical soil types are sand, loamy sand or sandy loam types of soils. Such soils have low runoff potential and high infiltration rates even when thoroughly wetted. They consist chiefly of deep, well to excessively drained sands or gravels and have a high rate of water transmission.
- Group B: Typical soil types are silt loam or loam. Such soils have a moderate infiltration rate when thoroughly wetted and consists chiefly or moderately deep to deep, moderately well to well drained soils with moderately fine to moderately coarse textures.
- Group C: Typical soil type is sandy clay loam. Such soils have low infiltration rates when thoroughly wetted and consist chiefly of soils with a layer that impedes downward movement of water and soils with moderately fine to fine structure.
- Group D: Typical soil types are clay loam, silty clay loam, sandy clay, silty clay or clay. Such soils have the highest runoff potential, as they have very low infiltration rates when thoroughly wetted and consist chiefly of clay soils with a high swelling potential, soils with a permanent high water table, soils with a clay pan or clay layer at or near the surface and shallow soils over nearly impervious material.

Moreover, SCS classifies three major classes of LU/LC as urban, cultivated, and woods and forests. These classes were further categorized into various subclasses, on the basis of land treatment practices such as contoured, terraced, straight row, bare, etc. (Chow *et al.*, 1988).

5.3.3 Shortcomings of classical CN estimations

Recently, Savvidou *et al.* (2018) discussed a number of shortcomings of the standard CN method, taking into account literature references as well as the gained experience from the research project DEUCALION. The project has been elaborated the research team ITIA during 2011-2014 and involved, among others, the analysis of numerous flood events in pilot catchments across Greece and Cyprus, using the NRCS-CN approach (for detailed description please refer to Efstratiadis *et al.*, 2014).

An important deficiency of the standard approach for CN extraction is the ignorance of the effect of slope on flood runoff generation. In fact, the reference CN values provided in the standard SCS tables were mainly identified from small agricultural watersheds with mild slopes, considering that the rainfall-runoff transformation is only affected by the soil and land cover characteristics. However, in the general case, the relief characteristics also affect the hydrological response of a watershed. In particular, steep slopes cause reduction of initial abstractions, decrease in infiltration, and reduction of the recession time of overland flow,

which in turn results in increased surface runoff (Montgomery and Dietrich, 2002). Currently it is generally accepted that the reference CN values are applicable for terrain slopes around 5%, and several researchers have proposed empirical formulae for adjusting the CN-values to slope (Huang *et al.*, 2006; Xu *et al.*, 2011; Deshmukh *et al.*, 2013; Verma *et al.*, 2017).

Moreover, the classification of soil types does not cover the entire range of permeability characteristics of the geological formations that are dominant in several areas worldwide. For example, a number of Mediterranean watersheds lie in highly permeable terrain (e.g., limestone, dolomite, karst), thus resulting in very low runoff rates (Merheb *et al.*, 2016). Yet, according to the typical classification by SCS, these should be classified in group A, representing sand, loamy sand or sandy loam types of soils. Reported experience with the use of the NRCS approach for flood estimations in such basins indicates that the associated CN values were quite overestimated; in fact, much lower values, of about 30 to 40, should be better employed to represent the significant infiltration losses (Efstratiadis *et al.*, 2014b).

Another difficulty with CN derivation from NRCS tabular data is the subjectivity involved in the determination of representative parameter values, through combining land cover classes and hydrological soil groups across different hydrological conditions. The estimations are based on qualitative information rather than on numerical criteria, while for several common cases the recommended values range too widely (particularly for soil types of category A). Therefore, quite different interpretations may be given for similar land cover and soil characteristics, thus resulting in significant uncertainty in the determination of CN values.

5.3.4 Revised method for CN assessment

Accounting for the aforementioned rationale, Efstratiadis *et al.* (2014a) proposed an analytical method for assessing the reference CN value over an area of interest, also facilitating spatial calculations in GIS environments (latter formalized by Savvidou *et al.*, 2018). In particular, the proposed classification is based on the categorization of three (instead of two) physiographic characteristics, each one comprising five classes, henceforth referred to as permeability, land use/cover, and drainage capacity. Indicative input geographical data for the production of the associated thematic layers in rural areas may include hydro-lithological or soil maps, land use/cover maps, terrain slope maps, and any other relevant information. In urban or suburban areas, information about building features may also be accommodated as any other relevant urban features.

Permeability classifications in rural areas account for the mechanical properties of the soil and the unsaturated zone (e.g., horizontal and vertical hydraulic conductivity) that affect infiltration, interflow and percolation mechanisms. Based on hydro-lithological or soil maps and depending on the predominant soil type underlying geological formation and structures (for urban or suburban areas), the permeability class is first described as very high, high, moderate, low or very low.

The density of structures, building features and open space development define the classification regarding the urban areas. A ranking from 1 to 5 is assigned, where index 1 refers to very high-permeability substrata (e.g., karst) and 5 to very low-permeability substrata (e.g., dense rocks). Residential areas range from class 3 to 5, according to their built density.

Vegetation classes are formulated on the basis of land characteristics related to retention mechanisms, soil roughness and filtration capacity, e.g. due to root zone growth. The vegetation class of the area is described as dense, moderate, undergrowth, sparse or zero. A ranking from 1 to 5 is assigned, where index 1 refers to dense vegetation class (e.g., evergreen forests) and 5 to bare soil. Savvidou *et al.* (2018) recommend that burned areas be classified under one

category with respect to their original condition; for instance, a burned coniferous forest should be classified as moderate vegetation class, thus assigning rank 2 instead of 1.

The geomorphological characteristics also play a key role in the drainage capacity of the area. These define the development of the river network and the existence of runoff regulation systems across the area of interest (e.g., land reclamation works, retention structures, sewer networks). The drainage capacity class is first described as negligible, low, moderate, high and very high, and then a ranking from 1 to 5 is assigned. In the absence of other information, these ranks may be exclusively assigned on the basis of five terrain slope categories, since this is an easily-retrieved property through typical DEM processing. In this respect, the first rank is assigned to horizontal areas, while the last rank is assigned to slopes over 30%.

The three soil classes, i.e. permeability, land use/cover and drainage capacity, are quantified through the corresponding indices, i_{PERM}, i_{VEG} and i_{SLOPE}, ranging from 1 to 5 (Table 1). Based on them, the representative value of CN is estimated through the empirical relationship:

$$CN = 10 + 9 \times i_{PERM} + 6 \times i_{VEG} + 3 \times i_{SLOPE} \quad \textbf{(5.4)}$$

According to the above formula, the minimum CN value is 28, while the maximum is 100. The former refers to the extreme case of areas with very high permeability, dense vegetation and negligible drainage capacity, while the latter is by definition applicable to areas that are permanently covered by water (rivers, lakes etc.), where all rainfall is converted to runoff. The three multipliers reflect the relative impacts of the corresponding physiographic characteristics to surface runoff generation. Considering only integer values for the three indices, the number of potential CN classes is 25 (given that different combinations of the three indices may result in the same CN value), while further classes can be identified by also allowing intermediate, non-integer values of the three indices. This empirical formula is actually compatible with the standard CN approach; for example, the smallest value by NRCS (CN = 30) is slightly smaller (CN = 28).

Table 5.2: Coding of physiographic characteristics for the estimation of parameter CN for reference conditions (AMC type II and initial abstraction ratio 20%).

Permeability class	i_{PERM}	**Vegetation class**	i_{VEG}	**Drainage capacity class**	i_{SLOPE}
Very high	1	Dense	1	Negligible	1
High	2	Moderate	2	Low	2
Moderate	3	Low	3	Moderate	3
Low	4	Sparse	4	High	4
Very low	5	Negligible	5	Very high	5

The quantification of the three individual components of CN allows its direct implementation in a GIS environment, which is very important in hydrological studies. The detailed tabular data by NRCS can be used in parallel to assign proper permeability, vegetation and drainage capacity classes over the area of interest.

The proposed methodology by Savvidou *et al.* (2018) is considered to be applicable in a grid cell, taking advantage GIS facilities. The input data for CN estimation are provided by means of raster data for the three aforementioned indices. Based on the CN values calculated for each cell of the reference surface, a raster map can be produced showing the spatial distribution of the CN parameter. Figure 1 illustrates the typical procedure for extracting a CN map in a GIS environment. The raster layers of permeability, vegetation density and slope indices, with values from 1 to 5, are overlaid, to produce a raster map of distributed values of CN for the reference area of interest.

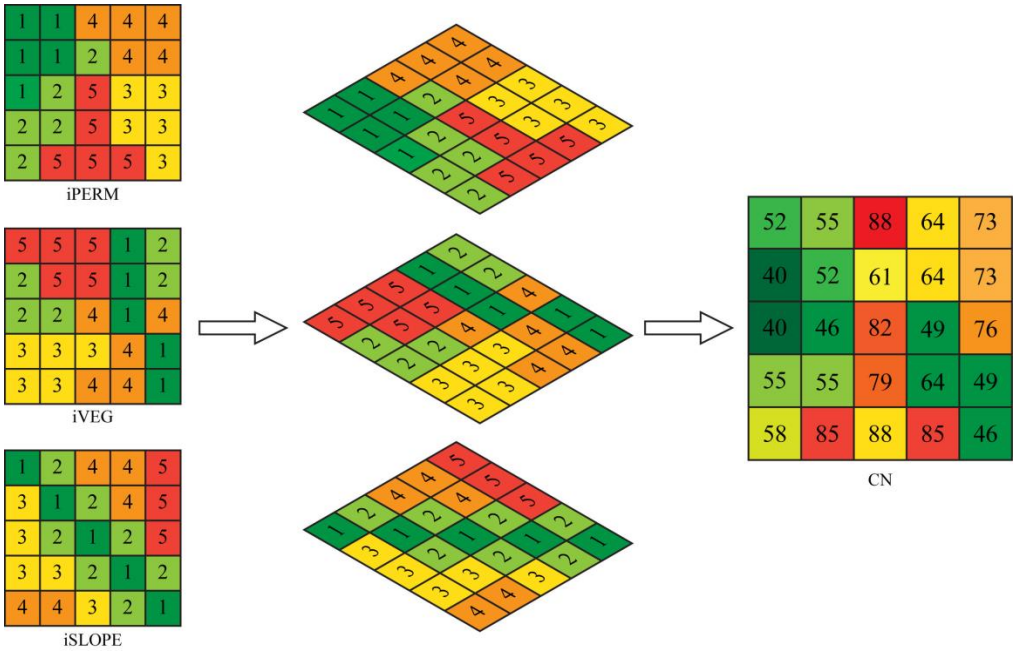


Figure 5.1: Layers of geographic information for permeability classes (i_{PERM}), vegetation density classes (i_{VEG}) and drainage capacity classes (i_{SLOPE}); (b) layer overlay; (c) CN parameter map (Savvidou *et al.*, 2018).

5.3.5 Adjustment to antecedent moisture conditions

The standard approach by SCS considers three antecedent soil moisture conditions classes (AMC I, AMC II and AMC III), depending on the total 5-day antecedent rainfall and the season category (dormant or growing). The three categories refer to dry, average or wet conditions, which statistically correspond to 90, 50, and 10% cumulative probability of exceedance of runoff depth for a given rainfall, respectively. For convenience, the CN values that are given in the literature (as well as the guideline documents by NRCS) refer to average conditions. For the other two AMC types, SCS uses the following conversion formulas, which are also plotted in Figure 5.2:

$$CN_I = \frac{4.2CN_{II}}{10 - 0.058CN_{II}} \tag{5.5}$$

$$CN_{III} = \frac{23CN_{II}}{10 + 0.13CN_{II}} \tag{5.6}$$

The antecedent soil moisture conditions are an important issue, which affect significantly the soil capacity characteristics at the beginning of the flood and, consequently, the surface runoff generation. In fact, the differences induced among the three typical antecedent moisture conditions of SCS are significant, particularly for CN values corresponding to high permeable or forested areas. For instance, as shown in Figure 5.2 for the reference (i.e., corresponding to AMC type II) value CN = 50, the adjusted values for dry and wet conditions are 30 and 70, respectively. In terms of potential maximum retention, the deviation is even larger, since the resulted values for AMC types I, II and III are 605, 254 and 110 mm, respectively.

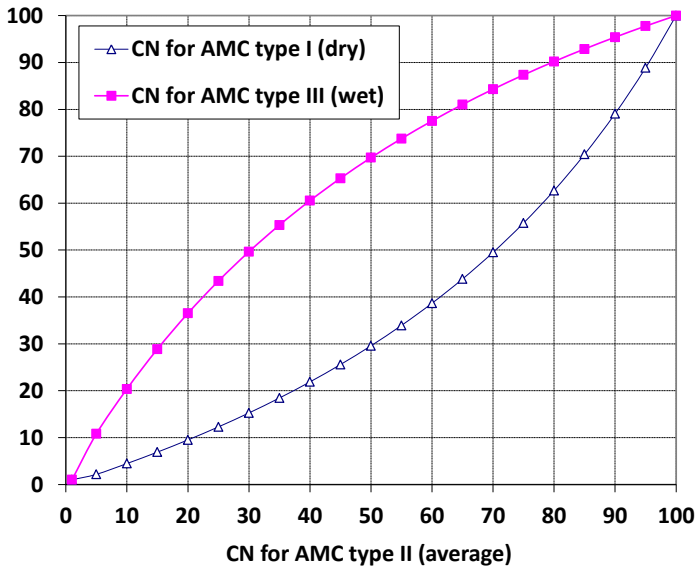


Figure 5.2: Plots of adjusted CN values for AMC types I and III, against the reference value, corresponding to AMC type II.

In this respect, the hypothesis of three discrete AMC types was revised, in order to better represent the inherent variability of the soil moisture, thus considering CN as a continuous variable, in order to implement a continuous instead of a discrete classification of antecedent moisture conditions we introduce a dimensionless parameter, symbolized AMC_{coef}. Assuming that 0.5 corresponds to Type II soil conditions, 0.1 corresponds to Type I and 0.9 corresponds to Type III, we can adjust the reference curve number to any AMC as follows:

$$CN_{AMC} = \begin{cases} CN_{II} - \dfrac{CN_{II} - CN_{I}}{0.4}\left(0.5 - AMC_{coef}\right), AMC_{coef} < 0.5 \\ CN_{III} + \dfrac{CN_{III} - CN_{II}}{0.4}\left(AMC_{coef} - 0.5\right), AMC_{coef} \geq 0.5 \end{cases} \quad (5.7)$$

It is also worth mentioning that in a recent investigation, based on extended statistical analysis of 5-day cumulative rainfall data in 215 stations over continental Greece and Crete, it was shown that the exceedance probability of each AMC type exhibit significant variability across continental Greece, following the significant variability of the hydroclimatic regime of Greece (Pontikos, 2014; Efstratiadis *et al.*, 2014a). This indicates that the percentiles of 10% and 90% that are assumed by SCS are not representative of the actual distribution of AMC, which is another shortcoming of the standard SCS approach. Nevertheless, by considering CN as continuous variable, it is possible to run the rainfall-runoff model for *any* initial soil moisture conditions, which makes the method much more flexible and realistic, as well.

5.3.6 Revisiting the standards for initial abstraction ratio estimation

As mentioned in section 5.3.1, the classical SCS approach recommends the use of a standard value for initial abstraction ratio, i.e. $\lambda = 0.20$. Actually, this assumption arises from analyses conducted in the late 60's, for establishing the curve number concept on the basis of rainfall-runoff observations at a number of experimental catchments. Within these experiments, SCS has analyzed numerous flood events, to extract the two parameters of the SCS method. In particular, regarding the initial abstraction ratio, while there was considerable scatter in the data, SCS reported that 50% of the examined events lay within the limits $0.095 \leq \lambda \leq 0.38$, which led adopting a standard value $\lambda = 0.20$.

However, several studies indicate that the actual range of variability of this parameter should be $0.0 \leq \lambda \leq 0.30$, while other studies have demonstrated that the initial abstraction is not constant, but it varies from storm to storm as well as from watershed to watershed (Ponce and Hawkins, 1996). For instance, by using model fitting methods to determine the ratio of h_{a0} to S for hundreds of rainfall-runoff events from numerous U.S. watersheds, Hawkins *et al.* (2002) concluded that a value of 0.05 rather than the commonly used of 0.20 would seem more appropriate. Thus conclusion was verified by Efstratiadis *et al.* (2014a), who analyzed numerous flood events across a number of catchments in Greece and Cyprus (within the aforementioned project DEUCALION). In most cases, the initial abstraction ratio λ values were around 0.05 or even smaller. This evidence is also supported by other researchers working in Mediterranean basins (Baltas *et al.*, 2007; Massari *et al.*, 2014).

5.3.7 Adjustment of maximum potential retention against initial abstraction ratio

The necessity for employing initial abstraction ratios that differ quite significantly from the standard value of 0.20 (typically, a quite smaller value should be applied), makes essential to adjust the reference CN values and associated maximum potential retention values, which have been extracted by considering $\lambda = 0.20$. Based on experimental results, Hawkins *et al.* (2002) proposed the use of the following adjusting formula, which is applicable for $\lambda = 0.05$:

$$S_5 = 1.33\, S_{20}^{1.15} \tag{5.8}$$

where S_{20} is the maximum potential retention estimated on the basis of reference CN, corresponding to $\lambda = 0.20$, while S_5 is the adjusted value to $\lambda = 0.05$. This adjustment denotes a change of rainfall-runoff transformation dynamics, by means of a quicker response of the basin (due to lower initial abstraction losses) yet lower generation of effective rainfall due to increased infiltration losses.

Here we propose a more generic approach, developed by Efstratiadis *et al.* (2014a), which is applicable to any value of λ. Key assumption is that for any value of λ, the effective rainfall, h_e, produced for a given gross rainfall, h, should equal the one provided by employing the reference curve number value, symbolized CN_{20}, for $\lambda = 20$. Under this premise, the following procedure is applied:

1. Estimate the maximum potential retention S_{20} corresponding to the reference CN, i.e.:
2. Compute the total effective rainfall h_e as function of h and S_{20}.

$$S_{20} = 254\left(^{100}/_{CN_{20}} - 1\right) \tag{5.9}$$

3. Solve the SCS formula inversely, to determine the maximum potential retention S_λ that correspond to the desirable initial abstraction ration λ. i.e.:

$$S_\lambda = \frac{2\lambda h + (1-\lambda)h_e - \sqrt{h_e[h_e(1-\lambda)^2 + 4\lambda h]}}{2\lambda^2} \tag{5.10}$$

Applying the above procedure cell-by-cell, we obtain the adjusted values of maximum potential retention for the given λ (common for the entire basin) as function of the spatially-varying gross rainfall h and reference CN. Next, we further adjust the corrected CN value against the antecedent soil moisture conditions.

5.4 Modified velocity approach for runoff propagation

5.4.1 Outline of the method

Output of the improved SCS-CN procedure it the spatiotemporal evolution of runoff across the basin and during the simulation period. In particular, at each cell we get a time series of effective rainfall values, to be propagated to the basin outlet.

Spatially-resolved runoff routing is typical problem of distributed hydrological models, usually tackled through time-area approaches, also referred to as isochronous methods, a brief overview of which is provided in the following section. Key requirement is the estimation of travel time, from each cell to the basin outlet. In general, this time comprises two components, i.e. a travel time over the terrain (overland flow) and a travel time along the river network (channel flow). For each component, different velocity models are applied that are explained I detail herein.

After determining the travel time of all cells, the basin is divided into a number of clusters, in order to employ the routing procedure in discrete time intervals. For convenience, in our approach the temporal resolution coincides to the resolution of the overall input of the model, i.e. the rainfall time series.

5.4.2 Isochronous method

The method of isochronous curves transforms the effective rainfall into a hydrograph, by calculating the time it takes for the water to reach the outlet from each geographic area of the basin. Essentially, the hydrograph is a transformation of the plot histogram that results in the output per time step. Initially, this method was mainly used to understand the drainage mechanism, but with the introduction of GIS it proved that it can adequately describe the phenomenon, thus providing reasonable hydrographs.

According to the classical configuration, the basin is divided into time zones, where the (effective) rainfall produced over each zone makes the same time to reach the outlet. As the rainfall is considered to be spatially uniform, the hydrograph at the outlet for the first time consists of the rainwater of the nearest zone, and then (and if the effective rain continues) more bands contribute to the total hydrograph.

Figure 5.3 shows the mechanism for creating the hydrograph, considering discreteness in four zones of equal size, in which the rainfall has a time equal to the concentration time and constant intensity. In particular, the basin is divided into n zones with areas A_1, A_2, ..., A_n, each one drained after time t = 1, 2, ..., n, respectively (Figure 5.4). If the total rainfall is equal to the accumulation time, with individual rainfall intensities i_1, i_2, ..., i_n, then the outlet runoff at each step is:

$$Q_n = i_n A_1 + i_{n-1} A_2 + \ldots + i_1 A_n \qquad \textbf{(5.11)}$$

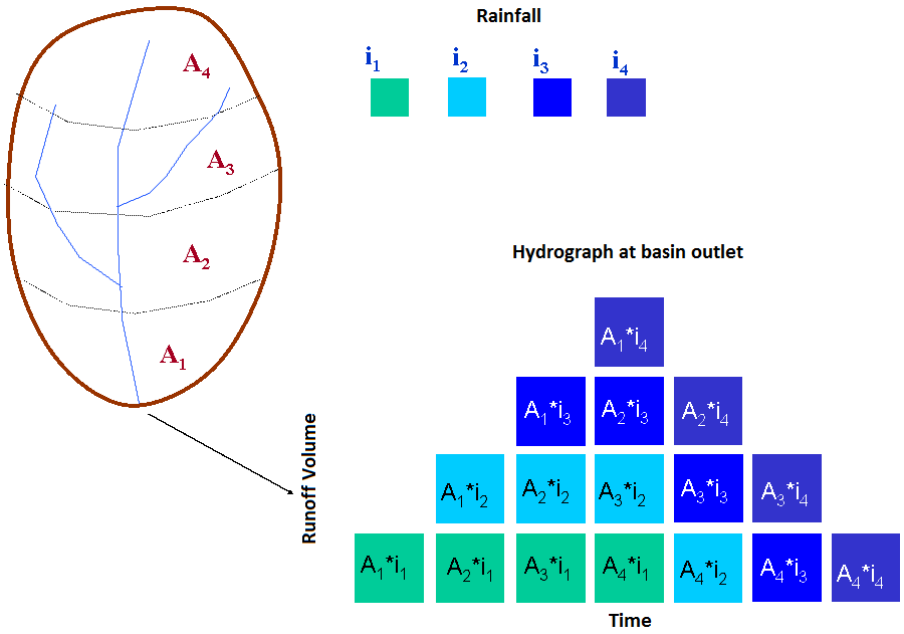


Figure 5.3: Example of the mechanism of hydrograph creation using the isochrones method, in a hypothetical basin of four zones of equal area with equal effective rainfall intensity.

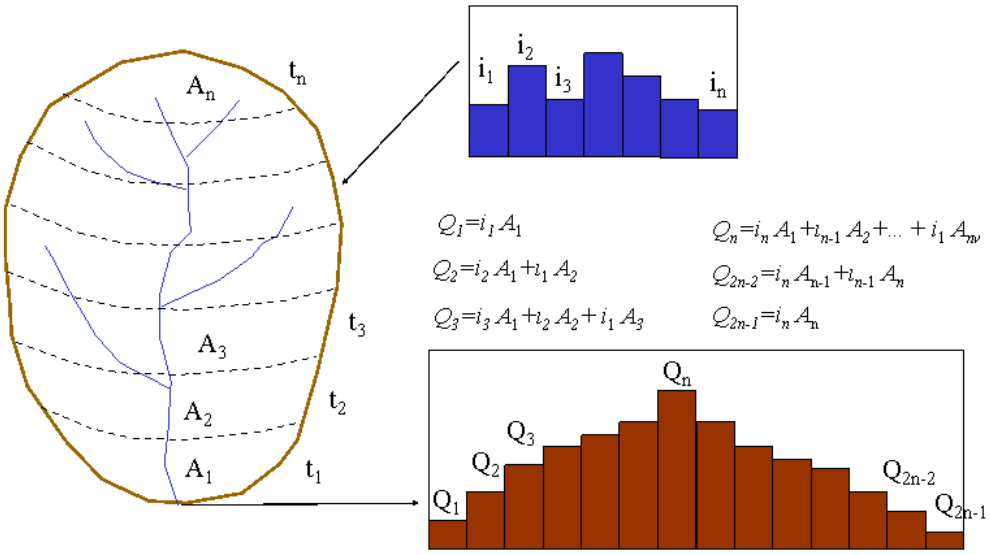


Figure 5.4: Expansion of the isochrones method to basins with zones of different area and rainfall intensity.

5.4.3 GIS implementation

With the introduction of GIS, the basin terrain is represented using the digital terrain model (DEM), that is a grid of the desired dimension is formed, separating the basin from tiles of known altitude. In the literature many techniques have been developed to calculate the runoff time of each cell up to the outlet of the basin. This time depends on: (a) the length of the path that follows the water incident on each cell to the outlet, and (b) the velocities of the water in each cell from those that it encounters until reaching the outlet.

The velocity on each cell depends on the following geographical factors:

- whether the cell is a land surface or belongs to the hydrographic network;
- the slope of the ground (or river);

- land characteristics and other factors that affect roughness.

Generally, the flow velocity is significantly increased across the hydrographic network, with the exception of very steep slopes and some land use categories. For this reason, in most applications, the hydrographic network is first identified and then the velocities, on the terrestrial surface (terrestrial flow) and the hydrographic network, are separately assessed. In this way, each cell corresponds to a path to the ground surface until it encounters the hydrographic network, which then follows until it reaches the outlet. Usually, land-flow velocities are calculated as a function of slope and roughness (land-use parameter), while the velocities across the hydrographic network are determined empirically or analytically calculated, based on hydraulic simulation models.

5.4.4 Estimation of overland velocities

Due to the extended use of Geographic Information Systems (GIS) in hydrological studies analysis tools are focused in representing the processed DEM in a slope-grid cell spatial scale. In this context, the total basin concentration time is the sum of the individual times of all the cells that form the maximum water path in the basin.

According to the American TR-55 specifications by NRCS (1986), a digital model of the basin is formulated, based on which the path of the water is drawn. Each cell along the path is identified either as a hillslope or as a channel. The velocity in the slope-type cells, i.e. the overland velocity is estimated by:

$$V_o = k\, J^{1/2} \tag{5.12}$$

where J is the slope of the cell calculated by typical GIS functions and k is a roughness coefficient. Haan *et al.* (1994) and little later MacCuen (1998) proposed k values for various land cover types, which are given in the Table 5.3. Note that the values are in the metric system (ft/s) and are converted to SI (m/s) by multiplying by 0.3048.

Table 5.3: Categories of land cover and proposed *k* values in ft/s (adapted from McCuen, 1998)

Land cover type	*k* (ft/s)	*k* (m/s)
Dense underbrush	0.7	0.2
Light underbrush	1.4	0.4
Heavy ground litter	2.5	0.8
Bermuda grass	1.0	0.3
Dense grass	1.5	0.5
Short grass	2.1	0.6
Short grass pasture	7.0	2.1
Conventional tillage with residue	1.2	0.4
Conventional tillage no residue	2.2	0.7
Agricultural, cultivated, straight row	9.1	2.8
Agricultural, cultivated, contour or strip cropped	4.6	1.4
Agricultural, trash fallow	4.5	1.4
Rangeland	1.3	0.4
Alluvial fans	10.3	3.1
Grassed waterway	15.7	4.8
Small upland gullies	23.5	7.2
Paved area	20.8	6.3

Paved gutter	46.3	14.1

Literature references (e.g., Grimaldi *et al.*, 2012) suggest that if the slope of a cell is greater than 4%, the following correction formula of slope J should be applied, in order to avoid overestimations of velocity:

$$J' = 0.05247 + 0.06363S - 0.182\ e^{-62.38J} \quad \textbf{(5.13)}$$

This approach is based on a much complete theoretical context compared to empirical formulas. However, it depends on the accuracy of the digital model, but also on multiple uncertainties and errors of the automatic hydrographic networking procedures. For instance, it is well-known that the travel time value increases, as the analysis of the digital model improves (e.g., Pavlovic and Moglen, 2008).

5.4.5 Estimation of channel velocities

As mentioned, the channel velocity is typically much larger than the overland one. By definition, this is essentially a hydraulic quantity, depending on the channel geometry, its hydraulic characteristics as well as the discharge. In this respect, the velocity values across a river network are spatially varying, and they are also temporally varying, since discharge is also a varying quantity. However, most of known literature approaches ignore the physical interpretation of velocity, thus employing the oversimplified assumption of a spatiotemporally constant value, typically equal to 2 m/s. Surprisingly, even recent, state-of-the-art advances, still accept this hypothesis (e.g., Petroselli and Grimaldi, 2018).

In our approach, the velocity in "conductor" cells belonging to the hydrological network is estimated by a pseudo-hydraulic approach, in the sense that it accounts, even at an abstract manner, for both the variability of the river network characteristics (slope, roughness) as well as the variability of discharge, by means of the recently developed concept of the varying time of concentration (Michailidi *et al.*, 2018). In this respect, the input data for our methodology are:

- a graph-type schematization of the river network, as a set of interconnected channel segments and associated junctions;
- the longitudinal slope of each channel segment, computed by dividing the elevation difference of the upstream and downstream junctions to the channel length;
- the representative Manning's roughness coefficient of each segment;
- the representative time of concentration of the specific flood event.

The first step in this estimation is the definition of cells belonging to the network. In this manner a certain *threshold* value of the number of cells that ultimately flow through these remaining conductor cells is defined. The procedure is simple, as reclassifying the flow accumulation of cells with regard to the threshold, defines the hydrological network and the cells where overland flow prevails.

For steady uniform flow across each channel segment *i*, the velocity V_i is given by the well-known Manning's formula:

$$V_i = \frac{1}{n_i} R_i^{2/3}\ J_i^{1/2} \quad \textbf{(5.14)}$$

where J_i is the average bed slope of the segment, n_i is the roughness coefficient and R_i is the hydraulic radius, which is function of the geometrical characteristics of the channel section and the water depth, which is in turn function of discharge.

In our context, we substitute the hydraulic radius term by a constant, i.e.

$$R_i^{2/3} = c \tag{5.15}$$

where *c* is considered as a lumped parameter of the river network, which is associated with the average runoff intensity of the flood event to be simulated. The estimation of *c* is made as follows:

Let t_c be the time of concentration of the entire basin and t_u be the time of concentration of its most upstream sub-basin (hereafter referred to as entrance time). According to its common definition, t_c represents the travel time across the longest flow path of the basin, i.e. from the hydraulically most remote point to its outlet. This comprises two components and associated flow types:

- shallow overland flow across the most upstream sub-basin, and
- channel flow across the main watercourse of the river network.

In complex networks, the determination of the longest river course upstream of the outlet junction is done by adding the individual lengths across all alternative courses. Moreover, in the case of multiple sub-basins drained to the upstream junction, we selected the one exhibiting with the longest entrance time (detailed recommendations on the delineation of the longest flow path are given by Michailidi *et al.*, 2018). Nevertheless, the entrance time is by definition estimated assuming overland flow conditions, thus employing eq. (5.16) with the use of a representative k_u value for the upstream sub-basin, i.e.

$$t_u = \frac{L_u}{V_u} = \frac{L_u}{k_u J_u^{1/2}} \tag{5.16}$$

where L_u and J_u are the flow length and average slope of the most upstream sub-basin.

For given times t_c and t_u, their difference t_r represents the total travel time across the longest river course, i.e. from the outlet of the most upstream sub-basin to the overall outlet, i.e.

$$t_r = t_c - t_u \tag{5.17}$$

Given that the flow path is constituted by a set of *N* channels, then t_r is equal to the sum of individual travel times, i.e.

$$t_r = t_1 + t_2 + \cdots + t_N \tag{5.18}$$

Under steady uniform flow conditions, we get:

$$t_r = \frac{L_1}{V_1} + \frac{L_2}{V_2} + \cdots + \frac{L_N}{V_N} \tag{5.19}$$

where L_i and V_i the length and velocity of each individual segment *i*.

By substituting the approximate formula for channel velocity we get:

$$t_r = c\left(\frac{J_1^{1/2}}{n_1 L_1} + \frac{J_2^{1/2}}{n_2 L_2} + \cdots + \frac{J_N^{1/2}}{n_N L_N}\right) \tag{5.20}$$

In the above procedure, the sole unknown quantity is the lumped parameter *c* (proxy of the hydraulic radius term), which is estimated by:

$$c = \left(\frac{n_1 L_1}{J_1^{1/2}} + \frac{n_2 L_2}{J_2^{1/2}} + \cdots + \frac{n_N L_N}{J_N^{1/2}} \right) (t_c - t_u) \quad \textbf{(5.21)}$$

In this respect, the hydraulic radius parameter depends on geometrical (length, slope) and hydraulic (roughness) properties of the main watercourse and the upstream sub-basin, which are constants, as well as the time of concentration of the basin, t_c, which is handled as a varying quantity depending on runoff intensity (see next section).

After determining parameter *c*, we can easily assign velocity values to any element of the river network, either belonging to the main water course or not. Actually, for given length, slope and Manning's roughness coefficient, the travel time along each segment *i* is given by:

$$t_i = \frac{L_i}{V_i} = \frac{n_i \, L_i}{c \, J_i^{1/2}} \quad \textbf{(5.22)}$$

Key advantage, and at the same time substantial improvement, of the proposed methodology against running literature approaches is the assignment of different velocity values across the channel segments, which is consistent with the fundamental hydraulic theory. As this theory implies, the channel velocity is proportional to the square root of the channel slope and inversely proportional to the roughness coefficient. Moreover, the velocity obviously depends on flow, and in our modelling framework this dependence is explicitly accounted for through the concept of the varying time of concentration. As mentioned before, the estimation of t_c is based on an original kinematic method that has been recently introduced by Michailidi *et al.* (2018), as explained hereafter.

5.4.6 The varying time of concentration and its implementation

From the origins of hydrology, the time of concentration, t_c, has been generally handled as a constant quantity. For, most of the traditional empirical formulas (e.g. Giandotti, Kirpich, SCS) associate this time with lumped geomorphological characteristics of the catchment (e.g. area, slope, river length), thus ignoring the obvious dependence of the travel time on runoff, which is generated over the catchment and is next propagated along the river network. The evident impact of this clear paradox error is the underestimation of flood flows, particularly for intense flood events that produce significant surface runoff, thus resulting in significantly increased flow velocities and, consequently, greatly decreased travel times against usual events. It is remarked that a flow-dependent time of concentration is a significant facet of nonlinearity within rainfall-runoff transformation since the two quantities are interrelated (i.e. the flow depends on time and vice versa; cf. Efstratiadis *et al.*, 2014b).

Both theoretical proof and empirical evidence imply that t_c exhibits significant variability against flow, thus making its definition and estimation a hydrological paradox (Grimaldi *et al.*, 2012). On the other hand, in the literature are found quite many approaches for associating the time of concentration with runoff (or rainfall). In fact, early attempts appear even from the 40's (Izzard and Hicks, 1946), while several empirical relationships have been employed in practice, a summary of which is given by Michailidi *et al.* (2018). However, most of them are applicable for specific engineering purposes (e.g., small urban catchments) or they are site-specific. Nowadays, the large expansion of GIS tools also enabled the employment of flow velocity approaches at a grid scale, thus providing "physically" sound approaches that provide cell by cell estimations of velocities and travel times, for given runoff. However, such approaches are

subject to several complexities and they are generally very sensitive against scaling assumptions.

In order to provide a generic and easily applicable methodology, Michailidi *et al.* (2018) proposed a kinematic approach under the assumptions of the rational method, to provide a GIS-based procedure for estimating the travel time across a catchment's longest flow path, as function of a given runoff intensity, which is considered uniformly distributed over the entire basin. We remark that a preliminary development of this method, applied to the river basin of Nedontas (study area of this thesis, also), was made by Antoniadi (2016). The method was tested in 30 basins from Italy, Greece and Cyprus, with different with respect to the basin shape, extent (i.e., the sample contained areas ranging from 14 to 1813 km^2) and land cover characteristics, as well as river network geometry.

Based on the outcomes of the analytical methodology, Michailidi *et al.* (2018) fitted power-type formulas associating the time of concentration (in hours) as function of the runoff intensity, i_e (mm/h), produced over the basin, i.e.

$$t_c = t_0 \, i_e^{-\beta} \tag{5.23}$$

where t_0 (h) is the so-called unit time of concentration, i.e. the travel time across the longest flow path under runoff intensity equal to 1.0 mm/h, and β is a shape parameter. We remark that the theoretical upper bound of β is 0.40, which refers to shallow flow conditions. On the basis of these results, Michailidi *et al.* (2018) also provided regional relationships, expressing the parameters t_0 and β as functions of lumped catchment properties, i.e. basin area, slope and length of main water course, and average channel roughness and width.

Latter, Michailidi (2018) also showed that eq. (5.23) can be easily implemented within event-based flood modelling, by means of a dynamic unit hydrograph, the shape of which is adjusted at each time step, accounting for the effective rainfall, as estimated by the SCS-CN method. Preliminary outcomes of this novel approach have been reported by Michailidi *et al.* (2017).

In our context, the implementation of the varying time of concentration (which is essential for estimating the hydraulic radius parameter, through eq. 5.21) is made in a more abstract manner, since the runoff routing is made analytically (cell-by-cell) and not via the unit hydrograph, which is a lumped conceptual model. Specifically, after determining the parameters t_0 and β, we assign a representative value of i_e. For convenience, we can directly employ the average intensity of the actual flood runoff (provided that the observed hydrograph is available). Alternatively, in case of missing flood data, we may consider a reasonable portion of the average rainfall intensity over the basin.

5.5 Enhanced model version for subsurface flow simulation

5.5.1 Rationale

By construction, the SCS-CN method only provides estimations of the surface (overland) runoff, by separating the effective rainfall from all rest hydrological deficits. However, it is widely accepted, even since the early 1960's (Hewlett, 1961; Betson, 1964; Hewlett and Hibbert, 1967), that particularly in many areas worldwide, where the infiltration capacity of soils is generally high in comparison with usual rainfall intensities (e.g. forested basins), the dominating component of a flood hydrograph is not the surface runoff one but the so-called *interflow*, also referred to as *throughflow* or *subsurface stormflow*. All these and similar terms are used to characterize the water draining from the soil either as unsaturated flow or, more

commonly, as shallow perched flow above the main groundwater level (Ward and Robinson, 1990, p. 200).

Based on real-world flood simulations with the standard SCS-CN procedure, Efstratiadis *et al.* (2014b) discussed the key shortcomings of this method, concluding that the most important one is its inability to produce interflow, because it does not include strictly a soil moisture balance. In fact, interflow is a rather smooth and slow flux (in contrast to overland flow, which generally follows the pattern of rainfall), and the dominant component of the recession process (i.e., the falling limb of the hydrograph). We note that this falling limb can be well represented as outflow from a linear reservoir (cf. Risva *et al.*, 2018).

Common experience suggests that in order to provide a physically – consistent formulation of the SCS-CN method, a radically different conceptualization is needed, to account for the soil moisture balance and the production of interflow during the time period of simulation. As will show herein, this was easily done, by introducing a soil moisture accounting component to the overall modelling procedure, and considering two additional fluxes, i.e. interflow and deep percolation, both being proportional to soil moisture storage.

For convenience, we first describe the lumped configuration of the enhanced, CN-based water balance model (in which the entire basin is represented as a single cell), and next explain its implementation within the distributed simulation scheme.

5.5.2 Lumped configuration

Key assumption of the water balance model is the treatment of maximum potential retention as varying quantity during the simulation period. This quantity, symbolized S_t, denotes the empty space of a conceptual tank of capacity K, employing the soil moisture accounting. In fact, as made in all common bucket-type conceptual hydrological models, this tank represents the unsaturated zone, which transforms the infiltrated rainfall into actual through the soil (upward vertical flux), interflow (horizontal flux) and percolation to deeper zones (downward vertical flux).

Due to the small time horizon of simulation, evapotranspiration processes are omitted (also because during a storm event the potential evapotranspiration is negligible), thus the water balance equation reads:

$$W_t = W_{t-1} + I_t - Y_t - G_t \tag{5.24}$$

where W_t is the soil moisture storage at time step t, and I_t is the infiltration, Y_t is the runoff produced through the soil (interflow), and G_t is the amount of water moving to deeper zones (i.e., the groundwater), and will be next transformed to baseflow and/or underground losses. Nevertheless, in our approach we assume that this transformation is very slow, thus allowing omitting the contribution of baseflow to the hydrograph, during the relatively short period of simulation.

In order to run the water balance model, it is essential determining the soil moisture storage at the beginning of simulation, i.e. W_0. By adding this quantity to the maximum potential retention for specific antecedent soil moisture conditions, symbolized S_0, we get the quantity:

$$K = W_0 + S_0 \tag{5.25}$$

which represents the storage capacity of the soil moisture accounting tank, which is by definition a constant threshold parameter. Under this premise, at the beginning of each time step, we can update the value of maximum potential retention, by substituting from the known capacity K the soil moisture storage so far, i.e.

$$S_t = K - W_{t-1} \quad \textbf{(5.26)}$$

The above assumption introduces a key difference with respect to the standard SCS-CN procedure, which handles the maximum potential retention as a constant during the evolution of flood event. In our approach, which is physically more consistent, the value of *S* changes, as additional fluxes are accounted for. In particular, *S* decreases as the inflow (i.e. infiltration) rate exceeds the outflow rate due to interflow and percolation, thus leaving less free space in the soil moisture tank. On the other hand, when rainfall stops, *S* is systematically increasing, since the tank is gradually getting empty. The concept of dynamically changing maximum potential retention introduces further nonlinearity to the rainfall-runoff transformation, and allows better describing the rising and falling limbs of the flood hydrograph.

Taking advantage of the governing equation of SCS-CN, we run the simulation with varying S_t, to estimate the surface runoff (effective rainfall), the initial abstractions (i.e. the amount of rainfall that is intercepted in the ground, without producing either runoff or infiltration) and the hydrological deficits, which consist the infiltration term of the water balance equation.

Interflow and percolation are considered as fractions of the soil moisture storage, i.e.:

$$Y_t = \kappa\, W_t \quad \textbf{(5.27)}$$

$$G_t = \mu\, W_t \quad \textbf{(5.28)}$$

where κ and μ are recession parameters.

As the rainfall-runoff transformation is implemented on a lumped basis, we also introduce a routing component to propagate the surface runoff to the basin outlet. This is implemented through a linear reservoir approach that implements of a lag-and-route scheme, formulated as:

$$Q_t = \varphi\, X_t \quad \textbf{(5.29)}$$

where X_t is the reservoir storage at time step *t*, and φ is a recession parameter. The reservoir storage is updated at the end of each time step as follows:

$$X_t = X_{t-1} + H_{et} - Q_t \quad \textbf{(5.30)}$$

where H_{et} is the effective rainfall produced at time step *t*, via the SCS-CN formula.

The total runoff is the sum of (routed) surface runoff and interflow. Both quantities arrive at the basin outlet with different hysteresis, expressed via time lag parameters δ, and τ, i.e.

$$R_t = Y_{t-\delta} + Q_{t-\tau} \quad \textbf{(5.31)}$$

Eventually, the above model contains five parameters, i.e.

- the initial abstraction ratio, λ, representing the upper threshold for abstraction losses as fraction of maximum potential retention;
- the AMC coefficient, which is next used to adjust the reference CN to any antecedent soil moisture conditions;
- the recession parameter, κ, controlling the generation of interflow;
- the recession parameter, μ, controlling the generation of percolation;
- the recession parameter φ, controlling the routing process;
- the lag time parameters, τ and δ.

The initial conditions of the model are expressed through two terms:

- the adjusted maximum potential retention, S_0, at the beginning of simulation.
- the soil moisture storage, W_0, at the beginning of simulation.

Overall input for employing the model is the reference runoff curve number, CN, which corresponds to 20% initial abstraction ratio and AMC type II. This allows for estimating the associated reference maximum potential retention, which is next adjusted to the given AMC and the given initial abstraction ratio, thus providing the essential initial condition, S_0.

5.5.3 Implementation within distributed simulations

Theoretically, in order to implement the aforementioned modelling approach in a distributed simulation context, it would be necessary to introduce numerous of additional parameters, to represent the interflow and percolation processes at each individual cell. This would result to a tremendously complex scheme that would be very difficult, if not impossible, to calibrate. Furthermore, the propagation of interflow through the soil would be subject to substantial uncertainty since, in contrast to surface runoff, the flow paths across the unsaturated zone are unknown.

For this reason, we finally developed a much simpler simulation scheme, by combining the enhanced SCS-CN approach for estimating the production of surface runoff and infiltration on a distributed basis, with a lumped approach for employing the soil moisture accounting in the unsaturated zone. In this vein, we added a lumped tank component to receive the distributed infiltration and transform it to interflow, percolation and change of soil moisture storage. We remark that key difference of the enhanced SCS-CN approach is the concept of the varying maximum potential retention within eq. (5.26).

Under these assumptions, the enhanced distributed model contains six lumped unknown quantities, namely:

- the initial abstraction ratio, λ;
- the AMC coefficient;
- the recession parameter, κ, controlling the generation of interflow;
- the recession parameter, μ, controlling the generation of percolation;
- the lag time parameter, δ, expressing the hysteresis of interflow;
- the initial soil moisture storage, W_0.

In fact, the AMC coefficient and the initial soil moisture storage are associated with the soil state at the beginning of simulation (hence they are not parameters but initial conditions), thus the sole parameters are the dimensionless quantities λ, κ and μ, and the time lag δ.

6 Model calibration

6.1 A general note on calibration of rainfall-runoff models

Once one or more models have been chosen for consideration in a project, it is necessary to address the problem of parameter calibration. It is not, in general, possible to estimate the parameters of models by either measurement or prior estimation. Studies that have attempted to do so have generally found that, even using an intensive series of measurements of parameter values, the results have not been entirely satisfactory (Beven *et al.*, 1984; Refsgaard and Knudsen, 1996; Loague and Kyriakidis, 1997). Prior estimation of feasible ranges of parameters also often results in ranges of predictions that are wide and may still not encompass the measured responses all of the time (Parkin *et al.*, 1996; Bathurst *et al.*, 2004).

There are two major reasons for the difficulties in calibration stage of a model. The first is that the scale of the measurement techniques available is generally much less than the scale at which parameter values are required. In general, however, obtaining the information required to use such a theory at the hillslope or catchment scale would be very time consuming and expensive and would result in a large number of holes in the hillslope. Thus, it may be necessary to accept that the small scale values that it is possible to measure and the effective values required at the model element scale are different quantities. The effective parameter values for a particular model structure still need to be calibrated in some way. It is also often the case that the time and space scales of model-predicted variables may be different from the scale at which variables of the same name can be measured (for example, soil water content). In this case, the variables used in calibration may also be incommensurate.

A common approach in calibration studies is the use of an optimization technique, so as to compare the results of repeated simulations with the available observations of the catchment response. The parameter values are adjusted between runs of the model, either manually by the modeler or by some computerized optimization algorithm until some “best fit” parameter set has been found. There have been many studies of optimization algorithms and measures of goodness of fit or objective functions in hydrological modelling.

The aim is to find the peak in the response surface in the parameter space defined by one or more objective functions. Figure 6.1 illustrates an example of the response surface for TOPMODEL. The two basal axes are two different parameter values, varied between specified maximum and minimum values. More specifically, the vertical axis is the value of an objective function, based on the sum of squared differences between observed and predicted discharges, that has the value 1 for a perfect fit. The visualization of a N – dimenitsional parameter hyperspace it is of course a more difficult task.

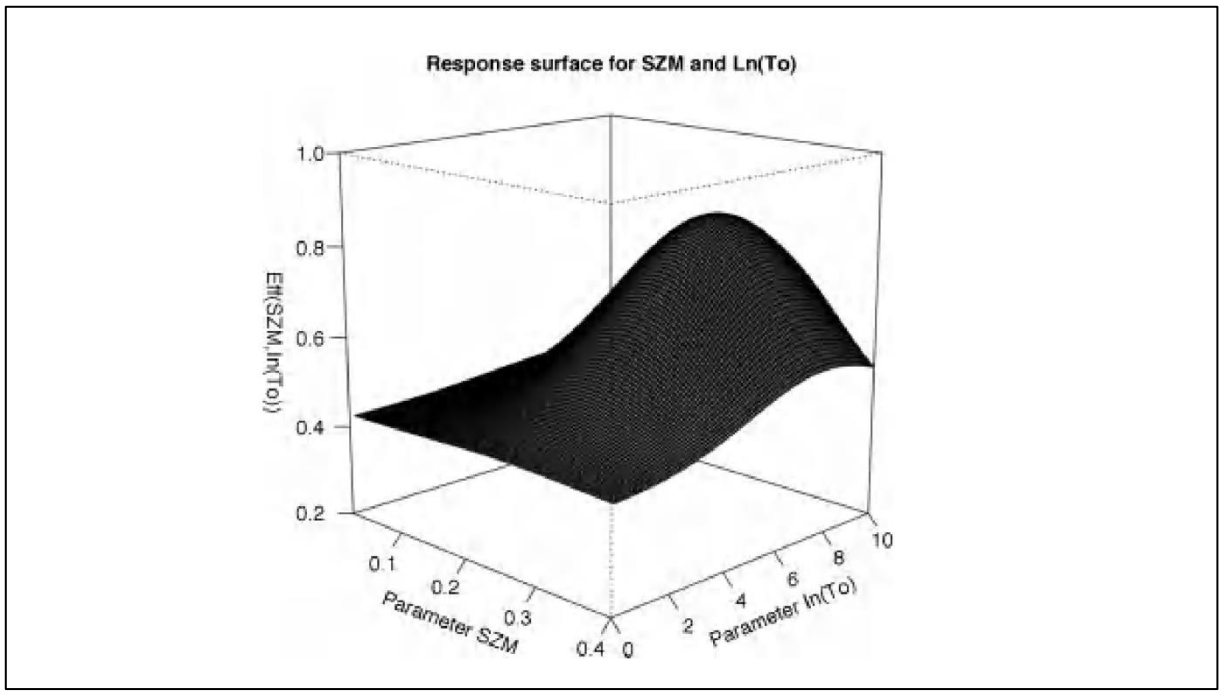


Figure 6.1: Response surface for two TOPMODEL parameters in an application to modelling the stream discharge of the small Slapton Wood catchment in Devon, UK; the objective function is the Nash – Sutcliffe efficiency that has a value of 1 for a perfect fit of the observed discharges (K.J.Beven, 2006).

Such surfaces can often be very complex and much of the research on optimisation algorithms has been concerned with finding algorithms that are robust with respect to the complexity of the surface in an N-dimensional space and find the global optimum set of parameter values. The complexity of the surface apart from the number of parameters, might also depend on the nature of the model equations, especially if there are thresholds involved, and the correct numerical integration of the equations in time (Kavetski and Clark, 2010).

However, for most hydrological modelling problems, the optimization problem is ill-posed in that if the optimisation is based on the comparison of observed and simulated discharges alone, there may not be enough information in the data to support the robust optimization of the parameter values (K.J Beven, 2006). Experience suggests that even a simple model with only four or five parameter values to be estimated may require at least 15 to 20 hydrographs for a reasonably robust calibration and, if there is strong seasonal variability in the storm responses, a longer period still (see, for example, Kirkby, 1975; Gupta and Sorooshian, 1985; Hornberger *et al.*, 1985; Yapo *et al.*, 1996). For more complex parameter sets, much more data and different types of data may be required for a robust optimization unless many of the parameters are fixed beforehand.

6.2 Differential evolution

6.2.1 Theoretical context

Optimization is the attempt to maximize a system's desirable properties, while minimizing its undesirable characteristics. However, what these properties are and how effectively they can be improved depends on the problem. In order to set an optimization problem, it is essential to define the objectives, the parameters and the constraints of the problem. An equation to be optimized given certain constraints and with variables that need to be minimized or maximized using nonlinear programming techniques is the objective function. For this study Differential Evolution Algorithm (DE) is selected.

Population Structure

It is the most versatile implementation maintains a pair of vector populations, both of which contain *Np D*-dimensional vectors of real-valued parameters. The current population, symbolized by P_x, is composed of those vectors, $x_{i,g}$, that have already been found to be acceptable either as initial points, or by comparison with other vectors:

$$P_{x,g} = (x_{i,g}), i = 0,1, \dots, N_p - 1, g = 0,1, \dots g_{max} \quad \textbf{(6.1)}$$

$$x_{i,g} = (x_{j,i,g}), j = 0,1, \dots, D - 1 \quad \textbf{(6.1)}$$

The index, g indicates the generation to which a vector belongs. Each vector is assigned a population index, i. Once initialized, DE starts to mutate, chosen vectors to produce an intermediary population, $P_{v,g}$ of N_p mutant vectors, $v_{i,g}$:

$$P_{v,g} = (v_{i,g}), i = 0,1, \dots N_p - 1, g = 0,1, \dots, g_{max} \quad \textbf{(6.3)}$$

$$u_{i,g} = (u_{j,i,g}), j = 0,1, \dots, D - 1 \quad \textbf{(6.4)}$$

Each vector in the current population is then recombined with a mutant to produce a trial population, P_u, of N_p trial vectors, $u_{i,g}$:

$$P_{u,g} = (u_{i,g}), i = 0,1, \dots, N_p - 1, g = 0,1, \dots, g_{max} \quad \textbf{(6.5)}$$

$$u_{i,g} = (u_{j,i,g}), j = 0,1, \dots, D - 1 \quad \textbf{(6.6)}$$

Initialization

It is essential, before the population is initialized, that both upper and lower bounds for each parameter must be specified. These 2*D* values can be collected into two, *D*-dimensional initialization vectors, b_L and b_U, for which subscripts L and U indicate the lower and upper bounds, respectively. Once initialization bounds have been specified, a random number generator assigns each parameter of every vector a value from within the prescribed range. For example, the initial value ($g = 0$) of the j^{th} parameter of the i^{th} vector is:

$$x_{j,i,0} = rand_j(0,1)(b_{j,U} - b_{j,L}) + b_{j,L} \quad \textbf{(6.7)}$$

The random number generator, $rand_j(0,1)$, returns a uniformly distributed random number from within the range [0,1), i.e., $0 \leq rand_j(0,1) < 1$. The subscript, j, indicates that a new random value is generated for each parameter. Even if a variable is discrete or integral, it should be initialized with a real value since DE internally treats all variables as floating-point values regardless of their type.

Mutation

Once initialized, DE mutates and recombines the population to produce a population of N_p trial vectors. In particular, differential mutation adds a scaled, randomly sampled, vector difference to a third vector. Equation (6.8) shows how to combine three different, randomly chosen vectors to create a mutant vector, $v_{i,g}$:

$$v_{i,g} = x_{r0,g} + F(x_{r1,g} - x_{r2,g}) \quad \textbf{(6.8)}$$

The scale factor, $F \in (0,1+)$, is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1.0. The base vector index, $r0$, can be determined in a variety of ways, but for now it is assumed to be a randomly chosen vector index that is different from the target vector index, i. Except for being distinct from each other and from both the base and target vector indices, the difference vector indices, $r1$ and $r2$, are also randomly selected once per mutant. Figure 6.2 illustrates how to construct the mutant, $v_{i,g}$, in a two-dimensional parameter space.

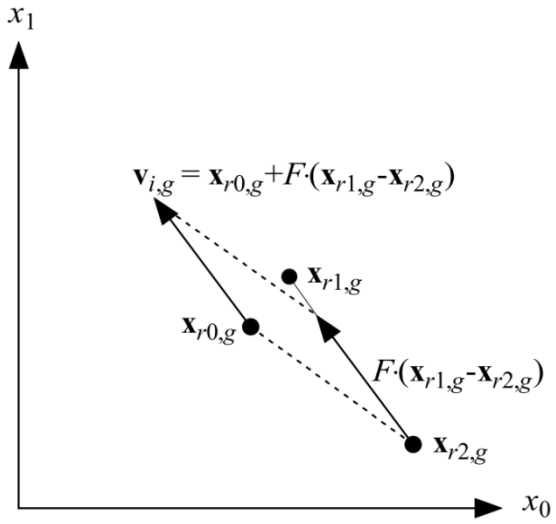


Figure 6.2: Differential mutation: the weighted differential, is added to the base vector, to produce a mutant.

Crossover

DE employs uniform crossover. Sometimes referred to as discrete recombination, (dual) crossover builds trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector:

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \; if \; (rand_j(0,1) \leq Cr \; or \; j = j_{rand} \\ x_{j,i,g} \; otherwise \end{cases} \quad \textbf{(6.9)}$$

The crossover probability, $Cr \in [0,1]$, is a user-defined value that controls the fraction of parameter values that are copied from the mutant. To determine which source contributes a given parameter, uniform crossover compares Cr to the output of a uniform random number generator, $rand_j(0,1)$. If the random number is less than or equal to Cr, the trial parameter is inherited from the mutant, $v_{i,g,}$ otherwise, the parameter is copied from the vector, $x_{i,g}$. In addition, the trial parameter with randomly chosen index, j_{rand}, is taken from the mutant to ensure that the trial vector does not duplicate $x_{i,g}$. Because of this additional demand, Cr only approximates the true probability, p_{Cr}, that a trial parameter will be inherited from the mutant. Figure 6.3 plots the possible trial vectors that can result from uniformly crossing a mutant vector, $v_{i,g}$, with the vector $x_{i,g}$.

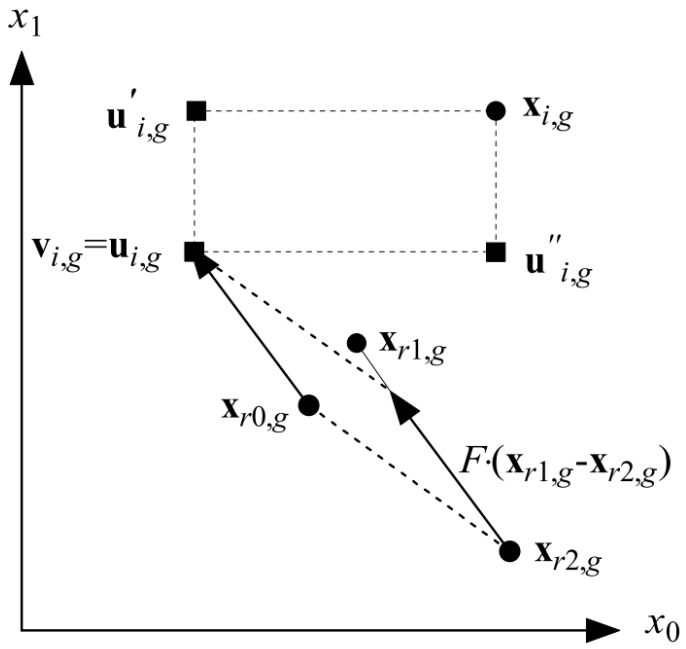


Figure 6.3: The possible additional trial vectors $\mathbf{u}'_{i,g}$, $\mathbf{u}''_{i,g}$ when $\mathbf{x}_{i,g}$ and $\mathbf{v}_{i,g}$ are uniformly crossed.

Selection

If the trial vector, $u_{i,g}$, has an equal or lower objective function value than that of its target vector, $x_{i,g}$, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation (6.10). By comparing each trial vector with the target vector from which it inherits parameters, DE more tightly integrates re-combination and selection than do other EAs:

$$x_{i,g+1} = \begin{cases} u_{i,g} & if\ \ f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} & otherwise \end{cases} \quad \textbf{(6.10)}$$

Once the new population is installed, the process of mutation, recombination and selection is repeated until the optimum is located, or a pre- specified termination criterion is satisfied, e.g., the number of generations reaches a preset maximum, g_{max}.

DE of scipy library, finds the global minimum of a multivariate function. Differential Evolution is stochastic in nature (does not use gradient methods) to find the minimum, and can search large areas of candidate space, but often requires larger numbers of function evaluations than conventional gradient based techniques.

There are several strategies for creating trial candidates, which suit some problems more than others. The 'best1bin' strategy is a good starting point for many systems. In this strategy two members of the population are randomly chosen. Their difference is used to mutate the best member (the best in best1bin), b_0, so far:

$$b' = b_0 + mutation * (population[rand0] - population[rand1]) \quad \textbf{(6.11)}$$

A trial vector is then constructed. Starting with a randomly chosen i^{th} parameter the trial is sequentially filled (in modulo) with parameters from b' or the original candidate. The choice of whether to use b', or the original candidate is made with a binomial distribution (the 'bin' in 'best1bin') - a random number in [0, 1) is generated. If this number is less than the recombination constant then the parameter is loaded from b', otherwise it is loaded from the

original candidate. The final parameter is always loaded from b'. Once the trial candidate is built its fitness is assessed. If the trial is better than the original candidate, then it takes its place. If it is also better than the best overall candidate, it also replaces that. To improve your chances of finding a global minimum use higher population size values, with higher mutation and (dithering), but lower recombination values. This has the effect of widening the search radius but slowing convergence.

6.2.2 Implementation of the differential evolution algorirthm

In order to apply the differential evolution algorithm, the sci.py library has been used. The **scipy.optimize.differential_evolution** uses the parameters of the Table below:

Table 6.1: Differential evolution parameters

Parameters	Description	Parameters	Description
Func	The objective function to be minimized. In the form f(x, *args), where x is the argument in the form of a 1-D array and args is a tuple of any additional fixed parameters needed to completely specify the function.	**Mutation**	Ranges between [0, 2]. Increasing the mutation constant increases the search radius, but will slow down convergence
Bounds	Boundsfor variables. (min, max) pairs for each element in x, defining the lower and upper bounds for the optimizing argument of *func*.	**recombination**	The recombination constant should be in the range [0, 1]. Increasing this value allows a larger number of mutants to progress into the next generation, but at the risk of population stability
Args	Any additional fixed parameters needed to completely specify the objective function.	**Seed**	*seed* for repeatable minimizations
strategy	The differential evolution strategy to use. For example: 'best1bin', 'best1exp','rand1exp'.	**disp**	Display status messages
Maxiter	The maximum number of times the entire population is evolved. The maximum number of function evaluations is: maxiter * popsize* len(x)	**Callback**	A function to follow the progress of the minimization

popsize	A multiplier for setting the total population size. The population has popsize * len(x) individuals.	**Polish**	If True (default), polishes the best population member at the end, which can improve the minimization slightly
tol	the solving process terminates when: convergence = mean(pop) * tol / stdev(pop) > 1	**init**	Specify how the population initialization is performed
		Res	The optimization result represented as an OptimizeResult object.

6.3 Performance Measures

A model requires a quantitative measure of performance or goodness of fit. In a hydrological model it is essential:

1. To predict the hydrograph peaks correctly (at least to within the magnitude of errors associated with the observations),
2. To predict the timing of the hydrograph peaks correctly,
3. To represent the form of the recession curve so as to set up the initial conditions prior to the next event.

6.3.1 Nash Sutcliffe Efficiency metric

A widely used goodness of fit measure based on the error variance is the modeling efficiency of Nash Sutcliffe (1970) defined as:

$$E = [1 - \frac{{\sigma_\varepsilon}^2}{{\sigma_0}^2}] \quad \textbf{(6.12)}$$

Where σ_0^2 is the variance of the observations and σ_ε^2 is the error variance, defined as:

$$\sigma_\varepsilon^2 = \frac{1}{T-1}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2 \quad \textbf{(6.13)}$$

Nash Sutcliffe efficiency metric (NSE) has the value of 1 for the perfect fit when is $\sigma_\varepsilon^2 = 0$; it has the value of 0 when is $\sigma_\varepsilon^2 = \sigma_0^2$, implying that the model g indicates that the model predictions are as accurate as the mean of the observed data. Negative values indicate that the model is performing worse than a 'no – knowledge' model, as the observed mean is a better predictor than the model. In other words, he residual variance (described by the numerator in eq. (6.13)), is larger than the data variance (described by the denominator).

6.3.2 Percent error in volume metric

The Percent Error in Volume (PEV) metric is defined as the percentage error of the total volume of the hydrograph, as the following equation is shown:

$$PEV = 100|\frac{V_0 - V_M}{V_0}| \quad \textbf{(6.14)}$$

where V_0 is the total volume of the observed hydrograph and V_m the total volume of the simulated hydrograph.

6.3.3 Percent error in peak flow

The Percent Error in Peak Flow (PEPF) is defined as the percentage peak error, without any relative temporal correlation between the observed and the simulated peak:

$$PERF = 100|\frac{Q_{0(PEAK)} - Q_{M(PEAK)}}{Q_{0(PEAK)}}| \quad \textbf{(6.15)}$$

6.3.4 Efficiency metric ΔT_{PF}

This index is the absolute time difference, expressed in minutes (min), between the observed and the simulated peak.

$$\Delta T_{PF} = |Tpeak_{obs} - Tpeak_{sim}| \quad \textbf{(6.15)}$$

7 Software implementation

In this chapter the model implementation and the procedure for creating a stand – alone program is discussed. Initially, the program is divided into two individual steps. A user interface class and the model implementation. The first one is responsible for the data handling, including a user interactive interface, whereas the latter one includes all the methods for data analysis.

It is worth noting, that two models have been developed with distinct purposes:

- Surface flow model, hereafter referred to as "surface model";
- Complete hydrograph model, also accounting for subsurface flow (interflow), hereafter referred to as "complete model".

All calculations are made in grid form and specifically raster files are used. Additionally some common GIS procedures that are available are extracted either by GRASS or by QGIS as part of preprocessing (these include the creation of the flow direction raster using a D8 scheme and the formulation of a stream network using the common hydrology toolboxes from these GIS tools). The various methods that are used in the two different models are thoroughly discussed in this chapter. These is also a plethora of secondary methods and classes which are not described here, in order to keep this chapter at a manageable size, enhance read-ability and retain the focus on the hydrological aspect of the thesis. Most of these secondary code snippets refer to packages'/libraries' actions, data handling within the software (like open file dialogs, combo-boxes, user-action buttons etc.) and visualization tools. The performance metrics' computation methods (NSE,PEV,$PEPF$, ΔT_{PF}) of section 6.3.1-4 are also skipped, as they are trivial in implementation and self-explanatory. We remark however, that these are implemented as methods of a Python class named *Metrics*.

7.1 Processing Functions

In order to make the program modular and easily extensible, we created a separated Python class that contains all the necessary methods for the model, denoted as *processingFuncs.py*. The most essential of them are analyzed in the following sections.

7.1.1 Flow accumulation

The first step is the creation of the flow accumulation grid. The flow accumulation method calculates the accumulated flow as the accumulated weight of all cells flowing into each downslope cell in the output raster. The value of cells in the output raster is the number of cells that flow into each cell. It is known, that cells with a high flow accumulation are areas of concentrated flow and may be used to identify stream channels. Cells with a flow accumulation of 0 are local topographic highs and may be used to identify ridges.

In this method the sole input parameter is the flow direction grid that is been imported from a raster data, computed in QGIS from the available DEM. In the following code snippet, the method is described. This method employs JIT (see 4.5.5) and runs in parallel mode (i.e. travel times from multiple cells are con-currently computed) to cut-down execution time.

```
@jit(parallel=True, nopython=False, nogil=True)
def flowaccumulation(flowdir):
    nr=flowdir.shape[0]
    nc=flowdir.shape[1]
    shape=(nr,nc)
```

```
    accumulation=np.zeros(shape)
    for i in prange(nr):
        for j in range(nc):
            if flowdir[i,j]==0:
               accumulation[i,j]=0
            else:
                tempi=i
                tempj=j
                while tempj!=-1 and tempj!=nc and tempi!=-1 and tempi!=nr
and flowdir[tempi,tempj]!=0:

                    if flowdir[tempi,tempj]==1:
                        movej=1
                        movei=0

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==16:
                            break
                    elif flowdir[tempi,tempj]==2:
                        movej=1
                        movei=1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==32:
                            break
                    elif flowdir[tempi,tempj]==4:
                        movej=0
                        movei=1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==64:
                            break
                    elif flowdir[tempi,tempj]==8:
                        movej=-1
                        movei=1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==128:
                            break
                    elif flowdir[tempi,tempj]==16:
                        movej=-1
                        movei=0

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==1:
                            break
                    elif flowdir[tempi,tempj]==32:
                        movej=-1
```

```
                        movei=-1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==2:
                            break
                    elif flowdir[tempi,tempj]==64:
                        movej=0
                        movei=-1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==4:
                            break
                    elif flowdir[tempi,tempj]==128:
                        movej=1
                        movei=-1

accumulation[tempi,tempj]=accumulation[tempi,tempj]+1
                        if flowdir[tempi+movei,tempj+movej]==8:
                            break
                    tempi=tempi+movei
                    tempj=tempj+movej
    return accumulation
```

Table 7.1: Input – output data of the flow accumulation method

Input data	Output
Flow direction raster	Flow accumulation raster

7.1.2 K raster creation

The following method creates the array based on the matching with the land uses codes according to the Corine land cover. The input data is in xls form and a dictionary is created for convenience. The outcome is a raster map, where every cell has a unique *k* value.

```
def createkraster(kdictionary,corinearray):
    import xlrd
    workbook=xlrd.open_workbook(kdictionary)
    sheet=workbook.sheet_by_index(0)
    nr=corinearray.shape[0]
    nc=corinearray.shape[1]
    shape=(nr,nc)
    karray=np.zeros(shape)
    data= [[sheet.cell_value(r, c) for c in range(sheet.ncols)] for r in
range(sheet.nrows)]

    def column(matrix, i):
        return [row[i] for row in matrix]
    corine_id=column(data,0)
```

```
    del corine_id[0]
    corine_id=[int(i) for i in corine_id]
    k=column(data,1)
    del k[0]
    d=dict(zip(corine_id, k))
    for i in range(nr):
        for j in range(nc):
            karray[i,j]=d.get(corinearray[i][j],0)
    return karray
```

Table 7.2: Input – output data of the *K* raster creation method

Input data	Output
K values dictionary from .xlsx file	K raster
Land cover values (CORINE) raster	

7.1.3 Slope raster file

In this method, a function from the gdal geospatial library is used to calculate the slope of each cell in the examined area. The format is chosen to be percent, as the following code snippet indicates.

```
def slope(dem):
    slopedem=gdal.DEMProcessing('slope.tif', dem, 'slope',
slopeFormat="-p")
    slopearray=slopedem.ReadAsArray()
    return slopearray
```

Table 7.3: Input – output data of the Slope raster creation method

Input data	Output
Digital Elevation Model raster	Slope (%)

7.1.4 Overland Velocity Grid

The next step, is the calculation of the overland velocity, based on the methodology discussed in the Chapter 5.

```
def velocityGrid(slope,karray,flowaccumulation,threshold):
    slope[slope==0]=0.001
    slope[slope<0]=0
    sqrt_slope=np.sqrt(slope)
    for i in range(slope.shape[0]):
        for j in range(slope.shape[1]):
            if slope[i,j]>0.04:
                slope[i,j]=0.05247+0.06363*slope[i,j]-0.182*np.exp
(-62.38*slope[i,j])
    velocity=np.multiply(karray, sqrt_slope)
    return velocity
```

Considering the slope (see 7.1.3) and flow accumulation grid (see 7.1.1), the k values and a threshold, the overland velocity is calculated (Chapter 4.4). The threshold defines the head sub-

basin in terms of area in pixels (e.g. for grid size of 25×25 m, and a threshold of 16000 cells, the minimum area of a defined basin is 10 km^2).

Table 7.4: Input – output data of the Overland Velocity method

Input data	Output
Slope raster	Overland Velocity raster
K values raster	
Flow accumulation raster	
Threshold in pixels	

7.1.5 Channel Velocity method

This method calculates the velocity in the stream network considering the observed discharge, the area of interest, the concentration time parameters t_u, t_0 and β, explained in detail within sections 5.4.5 and 5.4.6, as well as the time step of observations in seconds (e.g. for a 15-minute interval, the input is 900) and the start and finish time of the main hydrograph limbs to define the average runoff intensity (see section 5.4.6). Of course, the stream network object (defined from the streams class, see 7.1.10) is the key element in the calculation of the channel velocity, as this objects incorporates as attributes all the information needed (segment slope, Manning's coefficient, which segments are along the longest path etc.).

```
def Vchannel(Qobs,A,b,tA,t0,streams,streamr,Dt,start,end):
    T=(end-start)
    V=np.zeros(shape=T)
    for i in range(1,T):
        V[i]=((Qobs[i]+Qobs[i-1])/2)*Dt
    Vtotal=np.sum(V)/A/1000
    ieobs= Vtotal/(T/4)
    tc=t0*(ieobs**(-b))
    tR_h=tc-tA
    tR=tR_h*3600
    L=streams.LENGTH[streams.mainStreamLinks-1]
    sqrtJ=streams.SLOPE[streams.mainStreamLinks-1]**(0.5)
    n=streams.MANNING[streams.mainStreamLinks-1]
    b=streams.RiverB[streams.mainStreamLinks-1]
    c=np.sum((L*n)/(sqrtJ*b**0.5))/tR
    Vriver=(c*streams.SLOPE**(0.5)*streams.RiverB)/(streams.MANNING)
    return Vriver
```

Table 7.5: Input – output data of the Channel Velocity method

Input data	Output
Observed discharge timeseries	Channel Velocity
Area in km^2	
b parameter	

t0 parameter	
tA parameter	
Stream object	
Time base	
Start of hydrograph limb	
End of hydrograph limb	

7.1.6 Velocity Stack Method

The Velocity Stack method is essential for assigning cell velocities. This method overlays the channel velocity raster on top of the overland velocity raster, so as the overland cells that are also superimposed on the channel network are assigned only channel velocities. The outcome is the final velocity grid, used in the calculations travel time per cell to the basin outlet, which are next used to formulate the isochrones.

```
def VelocityStack(Ovelocity,Cvelocity,streamr):
    Tvelocity=np.zeros_like(Ovelocity)
    for i in range(Ovelocity.shape[0]):
        for j in range(Ovelocity.shape[1]):
            Tvelocity[i,j]=Ovelocity[i,j]
    for k in range(len(Cvelocity)):
        Tvelocity[np.where(streamr==k+1)]=Cvelocity[k]
    return Tvelocity
```

Table 7.6: Input – output data of the Velocity Stack method

Input data	Output
Overland velocity raster	Final Velocity
Channel velocity array	
Streams represented as raster	

7.1.7 Flow time

Flow time method calculates travel time of each cell to the outlet of the basin. In this study the isochronous curves are used as described in the Chapter 5.4.2. The parameters needed are the flow direction grid, the final velocity grid, the cell size in meters and the time step in seconds. This method employs JIT and runs in parallel mode (i.e. travel times from multiple cells are con-currently computed) to cut-down execution time.

```
@jit((numba.uint8[:,:],numba.float64[:,:],numba.float64,numba.float64),pa
rallel=True, nopython=False, nogil=True)
def flowtime(flowdir, velocity, cellsize, Dt):
    lx=cellsize
    lxy=(2*cellsize**2)**0.5
    nr=flowdir.shape[0]
```

```
    nc=flowdir.shape[1]
    shape=(nr,nc)
    accumtime=np.zeros(shape)
    for i in prange(nr):
        for j in range(nc):
            if flowdir[i,j]==0:
                accumtime[i,j]=0
            else:
                timeflow=0
                tempi=i
                tempj=j

                while tempj!=-1 and tempj!=nc and tempi!=-1 and tempi!=nr
and flowdir[tempi,tempj]!=0:
                    if flowdir[tempi,tempj]==1:
                        movej=1
                        movei=0
                        x=lx
                        if flowdir[tempi+movei,tempj+movej]==16:
                            timeflow=0
                            break
                    elif flowdir[tempi,tempj]==2:
                        movej=1
                        movei=1
                        x=lxy
                        if flowdir[tempi+movei,tempj+movej]==32:
                            timeflow=0
                            break
                    elif flowdir[tempi,tempj]==4:
                        movej=0
                        movei=1
                        x=lx
                        if flowdir[tempi+movei,tempj+movej]==64:
                            timeflow=0
                            break
                    elif flowdir[tempi,tempj]==8:
                        movej=-1
                        movei=1
                        x=lxy
                        if flowdir[tempi+movei,tempj+movej]==128:
                            timeflow=0
                            break
                    elif flowdir[tempi,tempj]==16:
                        movej=-1
                        movei=0
                        x=lx
                    elif flowdir[tempi,tempj]==32:
                        movej=-1
```

```
                    movei=-1
                    x=lxy
                elif flowdir[tempi,tempj]==64:
                    movej=0
                    movei=-1
                    x=lx
                elif flowdir[tempi,tempj]==128:
                    movej=1
                    movei=-1
                    x=lxy
                timeflow=timeflow+(x/velocity[tempi,tempj])

                tempi=tempi+movei
                tempj=tempj+movej

            accumtime[i,j]=timeflow/(Dt) #hours
    return accumtime
```

Table 7.7: Input – output data of the Flow time method

Input data	Output
Flow direction grid	Flow Time raster (values in timesteps)
Total velocity	
Cell size	
Flow direction grid	
tA	
Time base	

7.1.8 Isochrones creation

This simple method classifies the flow time raster into classes of 1 timestep, in order to create the isochrones areas. The cell value of the resulting raster refers to the isochrones' area by index.

```
def classifytimes(flowTIME):
    classtime=flowTIME//1
    return classtime
```

Table 7.8: Input – output data of the Flow time method

Input data	Output
Flow Time raster	Isochrone areas index raster

7.1.9 Inverse Distance Weighting (IDW)

This method uses the Inverse Distance Weighting (IDW) algorithm. This deterministic method for multivariate interpolation is used for the scattered rain gauges, so as to interpolate the

rainfall across the area. In this algorithm the assigned values to the unknown points of the area (in a grid base) are calculated with a weighted average of the values available at the known points. In a discrete assignment of the unknown function in a study region, the expected result is:

$$u(x): x \to \mathbb{R}, x \epsilon D \subset \mathbb{R}^n \tag{7.1}$$

where *D* is the study region, and the set of *N* data points can be described as a list of tuples:

$$[(x_1, u_1), (x_2, u_2), \dots, (x_N, u_N)] \tag{7.2}$$

The function aims to be "smooth" (continuous and differentiable), exact $(u(x_i) = u_i)$ and to meet the user's intuitive expectations about the phenomenon under investigation. Furthermore, the function is suitable for a computer application at a reasonable cost.

IDW is considered to be a very flexible spatial interpolation method, as the interpolation could be set in various ways.

Interpolated points are estimated based on their distance from known cell values. Points that are closer to known values will be more influenced than points that are farther away. A power of 1 smooths out the interpolated surface. A power of 2 increases the overall influence it has from the known values. It is worth noting that peaks and values are more localized and are not averaged out as much as with a power parameter of 1. The formula to calculate the value is:

$$z_p = \frac{\sum_{i=1}^{n}(\frac{z_i}{d_i^p})}{\sum_{i=1}^{n}(\frac{1}{d_i^p})} \tag{7.3}$$

The following figures illustrate some examples of the application of IDW algorithm.

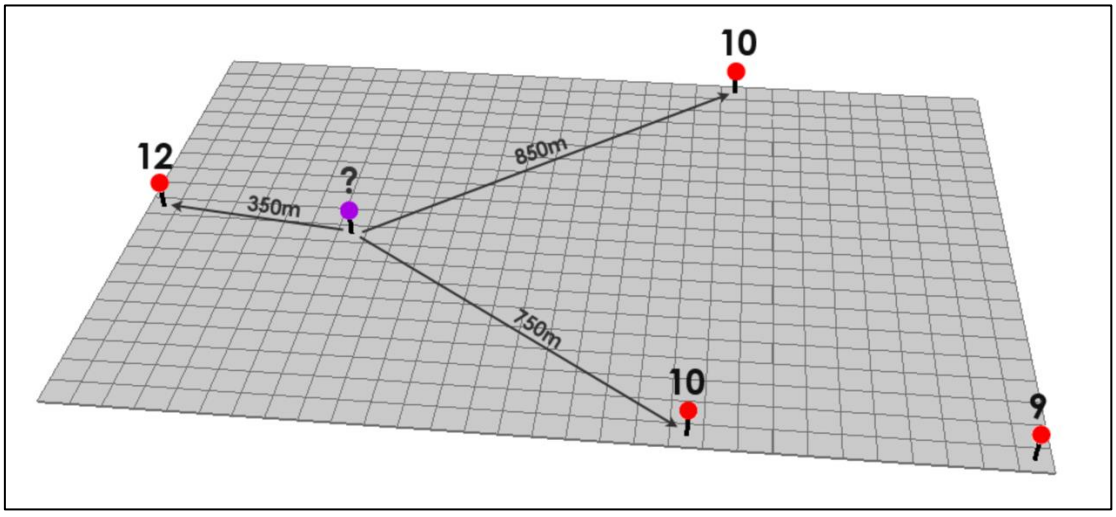


Figure 7.1: An example of the application of IDW (Source: gisgeography.com).

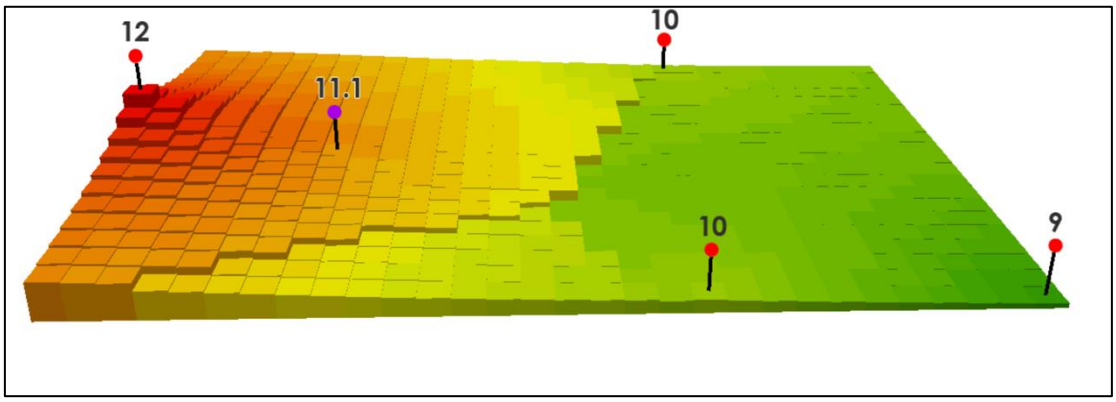


Figure 7.2: Example of IDW using power 1 (Source: gisgeography.com).

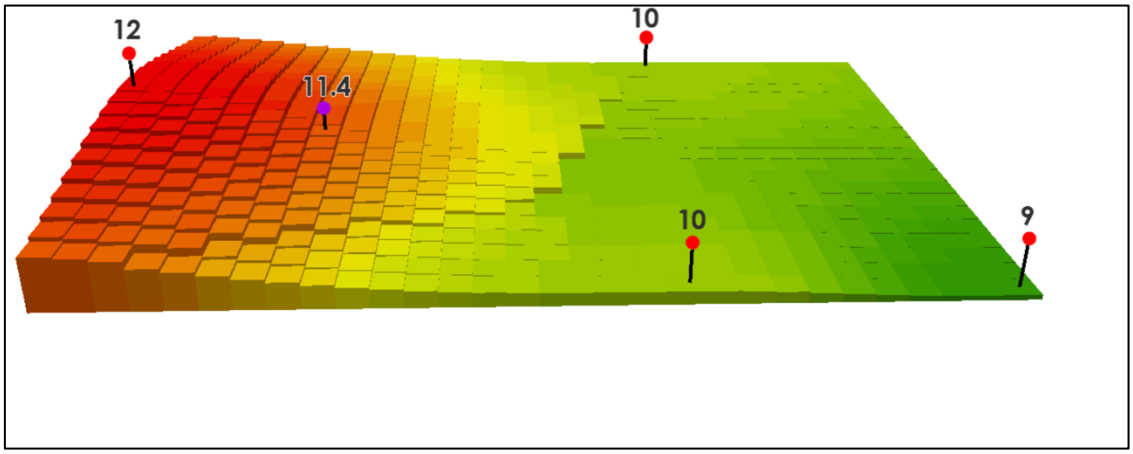


Figure 7.3: Example of IDW using power 2 (Source: gisgeography.com).

In the following code snippet, the implementation of the above interpolation is presented. The input parameters are the coordinates of the rain gauges, the size of the basin in terms of grid size and the cell size which defines the resolution of the raster Digital Elevation Model (DEM). In addition, the power parameter (suggested as 2.0) and the coordinates of the upper left corner of the raster are required. As a result, the distribution of the rainfall across the area is calculated.

```
def idw(stationX, stationY, rain, gridsize, xymin, power, cellsize):
    """ Calculates the idw of any number of stations for a rainfall timestep
        xystation: tuple with tuples (x,y) for stations
        rain: element with rainfall data of each station
        gridsize : tuple with dimensions [in cells] of raster
        power : the p of idw
        xymin : the upper left corner of raster in [m]
        cellsize : cell size in [m]
    """
    # get coordinates from tuple xymin
    xmin = xymin[0]
    ymin = xymin[1]
    # create the mesh
    x = np.arange(0, gridsize[1], 1)
    y = np.arange(gridsize[0]-1, -1, -1)
    xx, yy = np.meshgrid(x, y, sparse=False)
    # preallocate the distance 3D matrix
    weight =  np.zeros(shape=[gridsize[0], gridsize[1], len(stationY)])
```

```
    wu=np.zeros_like(weight)
    #nonzeroindex = np.zeros_like(distance)
    # loop for every station
    for s in range(len(stationX)):
        #stationXY = xystation[s]
        xstationtemp = stationX[s]
        ystationtemp = stationY[s]
        coorxx=xx * cellsize + xmin + cellsize / 2
        cooryy=yy * cellsize + ymin + cellsize / 2
        distance = np.sqrt((coorxx - xstationtemp)**2 + (cooryy-
ystationtemp)**2)
        checkifzero = np.nonzero(distance == 0)
        if not all(checkifzero): #check if it is not empty
            weight[:,:,s] = 1/distance**power
            wu[:,:,s] = weight[:,:,s]*rain[s]
        else:
            g = distance.flatten('F') # vectorize
            indexzero = np.nonzero(g == 0)
            weightvector=np.concatenate((1/g[:int(indexzero[0])]**
power,np.array([1]), 1/g[int(indexzero[0])+1:]**power))
            weight[:,:,s]=weightvector.reshape(gridsize[0],
gridsize[1],order='F').copy()
            wuv=np.concatenate(((1/g[:int(indexzero[0])]**power)*
rain[s],np.array([rain[s]]),(1/g[int(indexzero[0])+1:]**power)*rain[s]))

wu[:,:,s]=wuv.reshape(gridsize[0],gridsize[1],order='F').copy()
    idwrain=wu.sum(axis=2)/weight.sum(axis=2)
    return idwrain
```

Table 7.9: Inputs – outputs of the IDW method.

Input data	Output data
X, Y of the stations in tuple format	3D Matrix of spatial distribution of rainfall event : XY axes cell coordinates, Z axis time, cell value equals rainfall intensity
Rainfall timeseries	
Dimensions of the raster fie in pixels	
Power parameter	
Xmin, Ymin values in m (from the coordinate system	
Cell size	

7.1.10 Streamdata class

The streams class is used to instantiate a stream network object with its attributes as read by a shapefile.

```
class streamdata(object):
    def __init__(self, datastream):
```

```
        self.ID = np.array(datastream["ARCID"])
        self.SLOPE = np.array(datastream["SLOPE"])
        self.LENGTH=np.array(datastream["Length"])
        self.mainStreamLinks=np.array(datastream["MainLinks"])
        self.MANNING=np.array(datastream["Manning"])
        self.WIDTH=np.array(datastream["Width"])
# Instantiate example where datastream variable holds a reference to a
streams shapefile (.shp): streams=streamdata(datastream)
```

7.1.11 CN adjustment to *AMC* parameter

CN is adjusted according to antecedent soil moisture conditions (refer to section 5.3.5). Inputs are the CN raster (created through GIS operations according to 5.3.4). The output is the CN raster of the specific event. This method also employs JIT and runs in parallel.

```
@jit((numba.float64,  numba.float64[:,:]),parallel=True,  nopython=False,
nogil=True)
def cnadjustment(AMC,cnarray):
    cnadj=cnarray
    if AMC < 0.5:
        for i in prange(cnarray.shape[0]):
            for j in range(cnarray.shape[1]):
                temp=4.2*cnarray[i,j]/(10-0.058*cnarray[i,j])
                cnadj[i,j]=cnarray[i,j]-(cnarray[i,j]-temp)/0.4*(0.5-AMC)
    if AMC > 0.5:
        for i in prange(cnarray.shape[0]):
            for j in range(cnarray.shape[1]):
                temp=23*cnarray[i,j]/(10+0.13*cnarray[i,j])
                cnadj[i,j]=cnarray[i,j]+(temp-cnarray[i,j])/0.4*(AMC     -
0.5)
    return cnadj
```

Table 7.10: Inputs – outputs of the CN adjustment method

Input data	Output data
CN raster	Adjusted CN raster
AMC parameter	

7.1.12 Effective rainfall computation method

In order to compute the effective rainfall in every time step of the event and in every cell, inputs required are: the initial abstraction parameter λ (here as "a", because the Greek letter is not supported as a Python variable name), the adjusted by AMC coefficient CN raster in order to calculate S_{20} (as explained in eq. (5.9) in section 5.3.7) and the spatially distributed rainfall raster. The method calculates S_λ in every cell using eq. (5.10) and then forms the 3D matrix containing the effective rainfall for every cell & time step using eq. (5.1).

```
@jit((numba.float64,numba.uint8[:,:],numba.float64[:,:]),parallel=True,
nopython=False, nogil=True)
def ie_rain(a,cnarray,gridrain):
    S20=254*((100/cnarray)-1)
```

```
    S20[S20<0]=10000
    h0=0.2*S20
    h=np.sum(gridrain,axis=2)
    he=np.zeros_like(h)
    for i in prange(h.shape[0]):
        for j in range(h.shape[1]):
            if h[i,j]>h0[i,j]:
                he[i,j]=(h[i,j]-h0[i,j])**2/(h[i,j]-h0[i,j]+S20[i,j])
            else:
                he[i,j]=0
    Sa=(2*a*h+(1-a)*he-np.sqrt(he*(he*(1-a)**2+4*a*h)))/(2*a**2)
    Sa[Sa<0]=0
    h=np.cumsum(gridrain,axis=2)
    he=np.zeros_like(h)
    h0=a*Sa
    for t in prange(h.shape[2]):
        for i in range(h0.shape[0]):
            for j in range(h0.shape[1]):
                if h[i,j,t]>h0[i,j]:
                    he[i,j,t]=((h[i,j,t]-h0[i,j])**2)/(h[i,j,t]-h0[i,j]+
Sa[i,j])
                else:
                    he[i,j,t]=0
    ie=np.diff(he,axis=2)
    ie=np.concatenate((he[:,:,0:1],ie),axis=2)
    return ie
```

Table 7.11: Inputs – outputs of the effective rainfall computation method

Input data	Output data
CN adjusted raster	Effective rainfall 3D matrix (cell value: intensity, XY axes: coordinates, Z axis: time
Initial abstraction ratio parameter λ	
Raster of spatial distribution of rainfall	

7.1.13 Volume of isochrones computation method

This function computes cell runoff volume from each isochrone area and the respective cell effective rainfall for each time step. A matrix is created, so as to enable matrix algebra operations for hydrograph creation.

```
@jit((numba.float64[:,:],numba.float64[:],numba.float64[:,:],numba.float6
4,
numba.float64[:,:,:]),parallel=True, nopython=False, nogil=True)
def Vmat(V,time,classtime,cellsize,ie):
    E=(cellsize**2)
    for k in prange(ie.shape[2]):
        for j in range(int(time.max())):
            temp2=(classtime==j+1)*(ie[:,:,k])
```

```
            V[j,k]=np.sum(temp2)*E
    return V
```

Table 7.12: Inputs – outputs of volume of isochrones computation method

Input data	Output data
Pre-allocated volume matrix	Volume from isochrones matrix
Isochrones index raster	
Cell size value	
Effective rainfall 3D matrix	

7.1.14 Hydrograph computation method

The hydrograph computation method employs the methodology described in section 5.4.2. to compute the hydrograph at the basin outlet. Units are m^3/s. Inputs required are the augmented Volume matrix (by expanding the X,Y axis of the matrix to account for concentration time of the basin), the time step in seconds and a pre-allocated matrix to store the computed hydrograph.

```
@jit((numba.float64[:],    numba.uint8[:,:],numba.float64),parallel=False,
nopython=False, nogil=False)
def qcalc(Q,V2,Dt):
    for  j in range(len(Q)):
        Q[j]=0
        check1=np.min([j,V2.shape[0]-1])
        t=np.arange(check1+1)
        check2=np.min([j,V2.shape[1]-1])
        i=np.arange(check2,-1,-1)
        for x in range(len(t)):
            Q[j]=Q[j]+V2[t[x],i[x]]
    Qsim=(Q*10**(-3) )/Dt
    return Qsim
```

Table 7.13: Inputs – outputs of volume of isochrones computation method

Input data	Output data
Pre-allocated runoff matrix	Simulated runoff hydrograph
Volume from isochrones matrix	
Time step in seconds	

7.1.15 Surface Model method

This method executes all related aforementioned methods in order to generate a simulated hydrograph by employing the simpler surface hydrological model discussed in Chapter 5.

```
def model(a,m,Tvelocity,gridrain,cellsize,flowdir, cnarray,Dt):
    flowTIME=flowtime(flowdir, Tvelocity, cellsize,Dt)
    classtime=classifytimes(flowTIME)
```

```
    cn=cnadjustment(m,np.float64(cnarray))  #it was m*cnarray
    ie=ie_rain(a,cn,gridrain)
    ie[ie<0]=0
    time=np.unique(classtime[classtime>0])
    V=np.zeros(shape=(int(time.max()),ie.shape[2]))
    V=Vmat(V,time,classtime,cellsize,ie)
    Q=np.zeros(shape=(V.shape[1]+V.shape[0]-1,1))
    ex1=np.zeros(shape=(Q.shape[0]-V.shape[0], V.shape[1]))
    ex2=np.zeros(shape=(Q.shape[0], Q.shape[0]-V.shape[1]))
    V1=np.concatenate((V,ex1),axis=0)
    V2=np.concatenate((V1,ex2),axis=1)
    Qsim=qcalc(Q,V2,Dt)
    return Qsim
```

7.1.16 Complete Model method

This method executes the aforementioned methods to generate the enhanced model described in section 5.5.1 to 5.5.3. Some key changes to the surface method of section 7.1.15 are the water balance equations that are implemented within the effective rainfall computation so the simpler effective rainfall method is not called in execution. There is also the assumption that if interflow is not zero at the beginning of the simulation, then all previous values before the first time step are equal to the first interflow value (e.g. if lag parameter δ is equal to 10, interflow values at steps $t_0 - 1$: $t_0 - 10$ are all equal to interflow at step t_0.

```
def modelComplete(a,m,L,B,W0,lag,Tvelocity,gridrain,cellsize,flowdir,
cnarray,Dt,dem):
    flowTIME=flowtime(flowdir, Tvelocity, cellsize,Dt)
    classtime=classifytimes(flowTIME)
    cn=cnadjustment(m,np.float64(cnarray))
    S20=254*((100/cn)-1)
    S20[S20<0]=10000
    h0=0.2*S20
    h=np.sum(gridrain,axis=2)
    he=np.zeros_like(h)
    for i in range(h.shape[0]):
        for j in range(h.shape[1]):
            if h[i,j]>h0[i,j]:
                he[i,j]=(h[i,j]-h0[i,j])**2/(h[i,j]-h0[i,j]+S20[i,j])
            else:
                he[i,j]=0
    Sa=(2*a*h+(1-a)*he-np.sqrt(he*(he*(1-a)**2+4*a*h)))/(2*a**2)
    Sa[Sa<0]=10000
    h0=a*S20
    time=np.unique(classtime[classtime>0])
    interflow=np.zeros([int(time.max())+gridrain.shape[2],1])
    QinteflowM3S=np.zeros([int(time.max())+gridrain.shape[2],1])
    W=np.ones_like(Tvelocity)*W0
    K=Sa+W0
    h=np.cumsum(gridrain,axis=2)
```

```
he=np.zeros([gridrain.shape[0],gridrain.shape[1],gridrain.shape[2]+1])
    ie=np.zeros([gridrain.shape[0],gridrain.shape[1],gridrain.shape[2]])
    interflow=np.zeros([gridrain.shape[0],gridrain.shape[1],
gridrain.shape[2]+int(time.max())])
    percolation=np.zeros([gridrain.shape[0],gridrain.shape[1],
gridrain.shape[2]])
    for t in range(1,ie.shape[2]+1):
        df=h[:,:,t-1]-h0
        df[df<0]=0
        he[:,:,t]=(df**2)/(df+Sa)*(dem>0).astype(float)
        he[he<0]=0
        ie[:,:,t-1]=he[:,:,t]-he[:,:,t-1]
        ie[ie<0]=0
        infiltration=gridrain[:,:,t-1]-ie[:,:,t-1]
        W=W+infiltration
        interflow[:,:,t-1]=W*L
        interflow[interflow<0]=0
        W=W-interflow[:,:,t-1]
        percolation[:,:,t-1]=W*B
        W=W-percolation[:,:,t-1]
        Sa=K-W
    V=np.zeros(shape=(int(time.max()),ie.shape[2]))
    V=Vmat(V,time,classtime,cellsize,ie)
    Q=np.zeros(shape=(V.shape[1]+V.shape[0]-1,1))
    ex1=np.zeros(shape=(Q.shape[0]-V.shape[0], V.shape[1]))
    ex2=np.zeros(shape=(Q.shape[0], Q.shape[0]-V.shape[1]))
    V1=np.concatenate((V,ex1),axis=0)
    V2=np.concatenate((V1,ex2),axis=1)
    Qsim=qcalc(Q,V2,Dt)
    for t in range(ie.shape[2],ie.shape[2]+int(time.max())):
        interflow[:,:,t]=W*L
        W=W-interflow[:,:,t]
        W=W*(1-B)
    # transform interflow to m3/s
    for t in range(interflow.shape[2]):

QinteflowM3S[t]=(((np.sum(interflow[:,:,t]*(dem>0).astype(float))*
cellsize**2)/1000)/Dt)
    # add lag
    qz=np.zeros([lag,1])
    qlag=np.ones([lag,1])*QinteflowM3S[0]
    Qsim=np.concatenate((Qsim,qz),axis=0)
    QinteflowM3S=np.concatenate((qlag,QinteflowM3S[0:-1]),axis=0)
    Qtot=Qsim+QinteflowM3S
    Q=np.hstack((Qsim,QinteflowM3S,Qtot))
    return Q
```

7.2 Optimization methods

The following methods are called within optimization procedures. NSE is the default metric used for the optimization of model parameters.

- Surface model differential evolution call with parameter bounds and arguments, where x is the optimized parameter values:

```
bounds=[(0.0001, 0.50),(0.01, 1)]
args=(Qobs,Tvelocity,gridrain,cellsize,flowdir, cnarray,Dt,dem)
results=differential_evolution(processingFuncs.optimFun2, bounds, args,
strategy='best1bin', maxiter=20, popsize=20, tol=0.01, mutation=(0.5, 1),
recombination=0.7, seed=None, callback=None, disp=True, polish=False,
init='latinhypercube', atol=0)
x=results.x
```

- Surface model's objective function:

```
def optimFun(x,*args):
    Qobs=args[0]
    Tvelocity=args[1]
    gridrain=args[2]
    cellsize=args[3]
    flowdir=args[4]
    cnarray=args[5]
    Dt=args[6]
    a=x[0]
    m=x[1]
    Qsim=model(a,m,Tvelocity,gridrain,cellsize,flowdir, cnarray,Dt)
    Qobs=np.asarray(Qobs)
    NSE=Metrics.NSE(Qsim, Qobs)
    NSEnegative=-NSE
    return NSEnegative
```

- Complete model differential evolution call with parameter bounds and arguments, where x is the optimized parameter values, note the rescaling of parameters in the objective function:

```
bounds=[(0.1, 1),(0.01, 0.3),(0,1),(0,1),(0,1),(0,1)]
args=(Qobs,Tvelocity,gridrain,cellsize,flowdir, cnarray,Dt,dem)
results=differential_evolution(processingFuncs.optimFunComplete, bounds,
args, strategy='best1bin', maxiter=15, popsize=6, tol=0.01, mutation=(0.5,
1), recombination=0.7, seed=None, callback=None, disp=True, polish=True,
init='latinhypercube', atol=0)
x=results.x
```

- Complete model's objective function:

```
def optimFunComplete(x,*args):
    Qobs=args[0]
```

```
    Tvelocity=args[1]
    gridrain=args[2]
    cellsize=args[3]
    flowdir=args[4]
    cnarray=args[5]
    Dt=args[6]
    dem=args[7]
    a=0.5*x[0]
    m=x[1]
    L=x[2]*0.1
    B=x[3]*0.1
    W0=x[4]*50
    lag=int(x[5]*200)

Qsim3=modelComplete(a,m,L,B,W0,lag,Tvelocity,gridrain,cellsize,flowdir,
cnarray,Dt,dem)
    Qsim=Qsim3[:,2]
    NSE= Metrics.NSE(Qsim, Qobs)
    NSEnegative=-NSE
    return NSEnegative
```

7.3 Software application

The developed software application is presented within this section. The application is responsible for data handling, visualization and the execution of the models described in Chapter 5. The GUI application is mainly visually designed using Qt Designer (part of Qt Framework). Then, the designer file (.ui) is imported into the python script using the PyQt bindings.

The requirement for this software application is Python 3.6 or latest version and the Anaconda distribution to handle all necessary packages. It operates in every major operating system (Windows, MacOS, all Linux distributions).

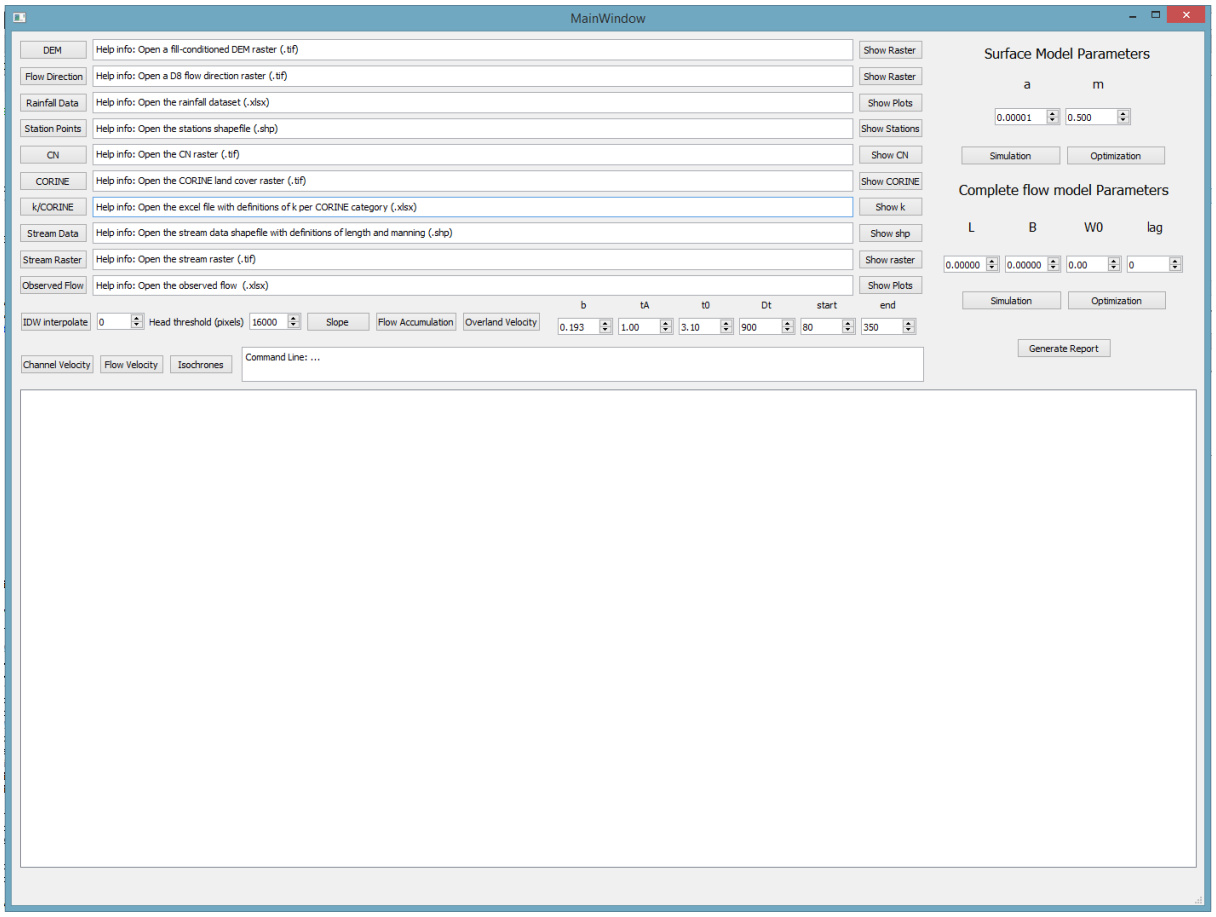


Figure 7.4: Overview of the application's main window.

As seen in Figure 7.4, the top left area of the window is the data input wizard, with visual and text guidance to import all necessary files. These include:

- Digital Elevation Model (.tiff)
- Flow Direction raster file (.tiff)
- Rainfall data (.xlsx)
- Gauges points (.shp)
- Curve Number raster file (.tiff)
- Corine land cover raster file (.tiff)
- The k values per Corine category (.xlsx)
- Stream length and Manning values (.shp)
- Stream raster file (.tiff)
- Observed flow (.xlsx)

When the user clicks in a data input (for example the DEM) a pop – up window opens makes it available to load only the files with the same form (e.g. only .tiffs in this case). In addition, above every data input at the main window a help info provides the necessary information for every action. There are also buttons on the right side of the wizard to visualize these inputs, at the window's graphics view widget. The command line widget displays important information about user actions. Examples are given in the following figures.

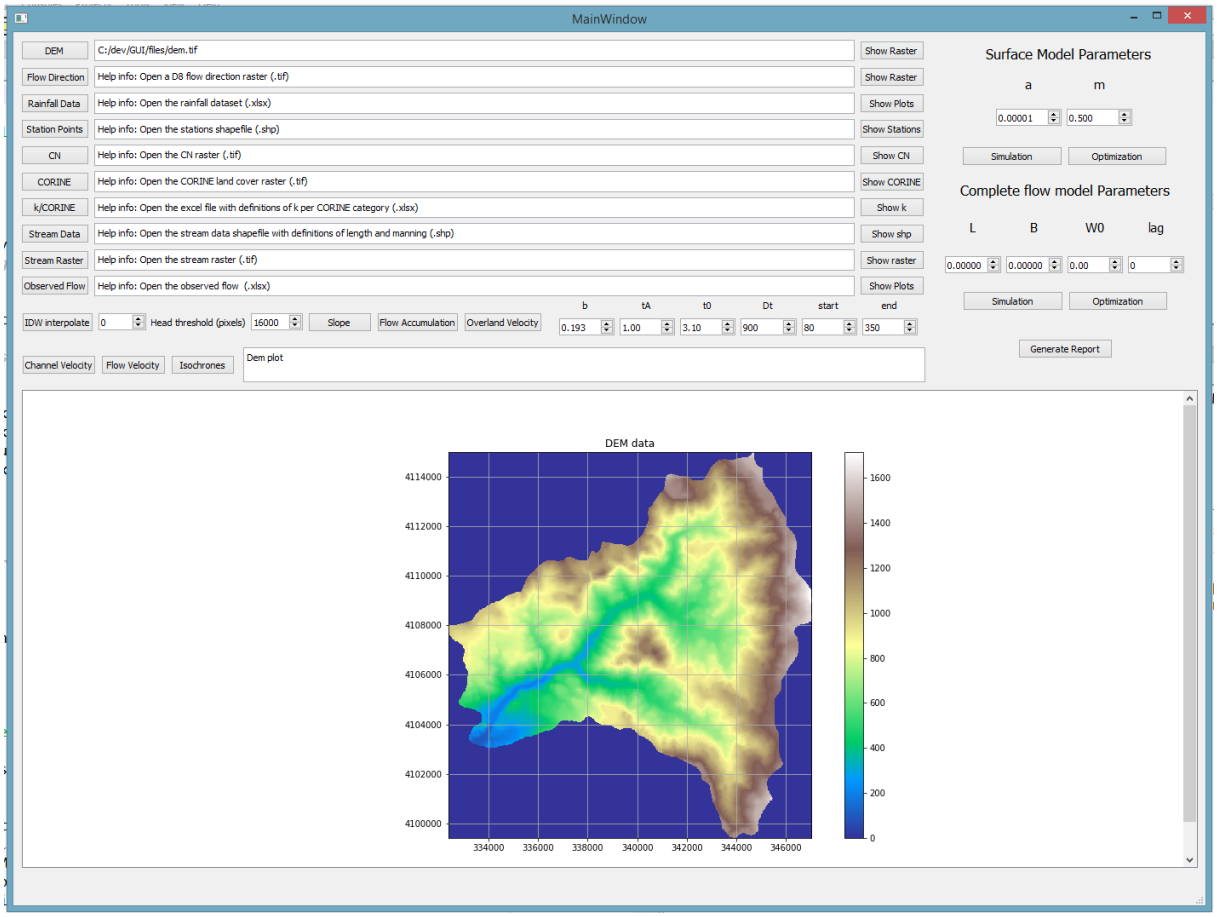


Figure 7.5: Example of DEM import.

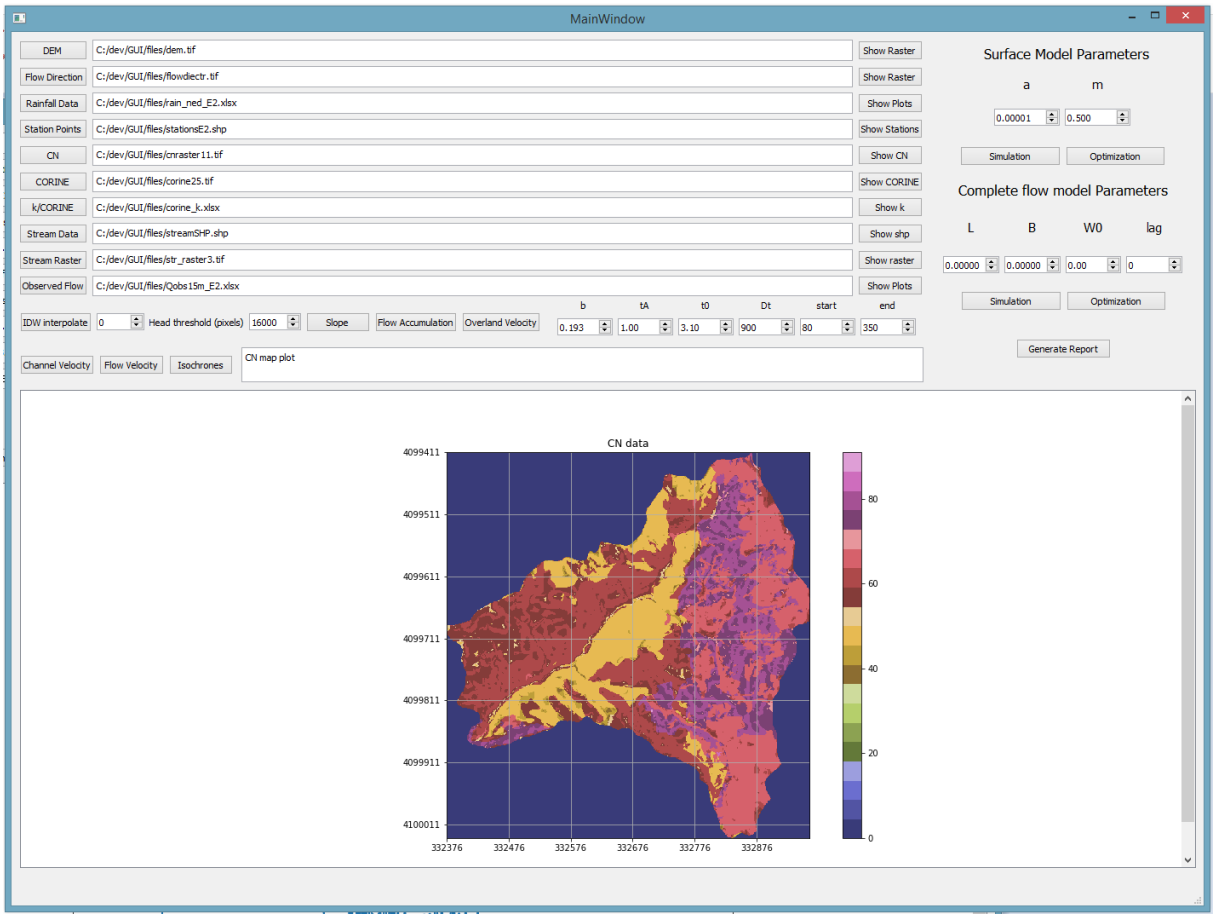


Figure 7.6: Example of CN raster import.

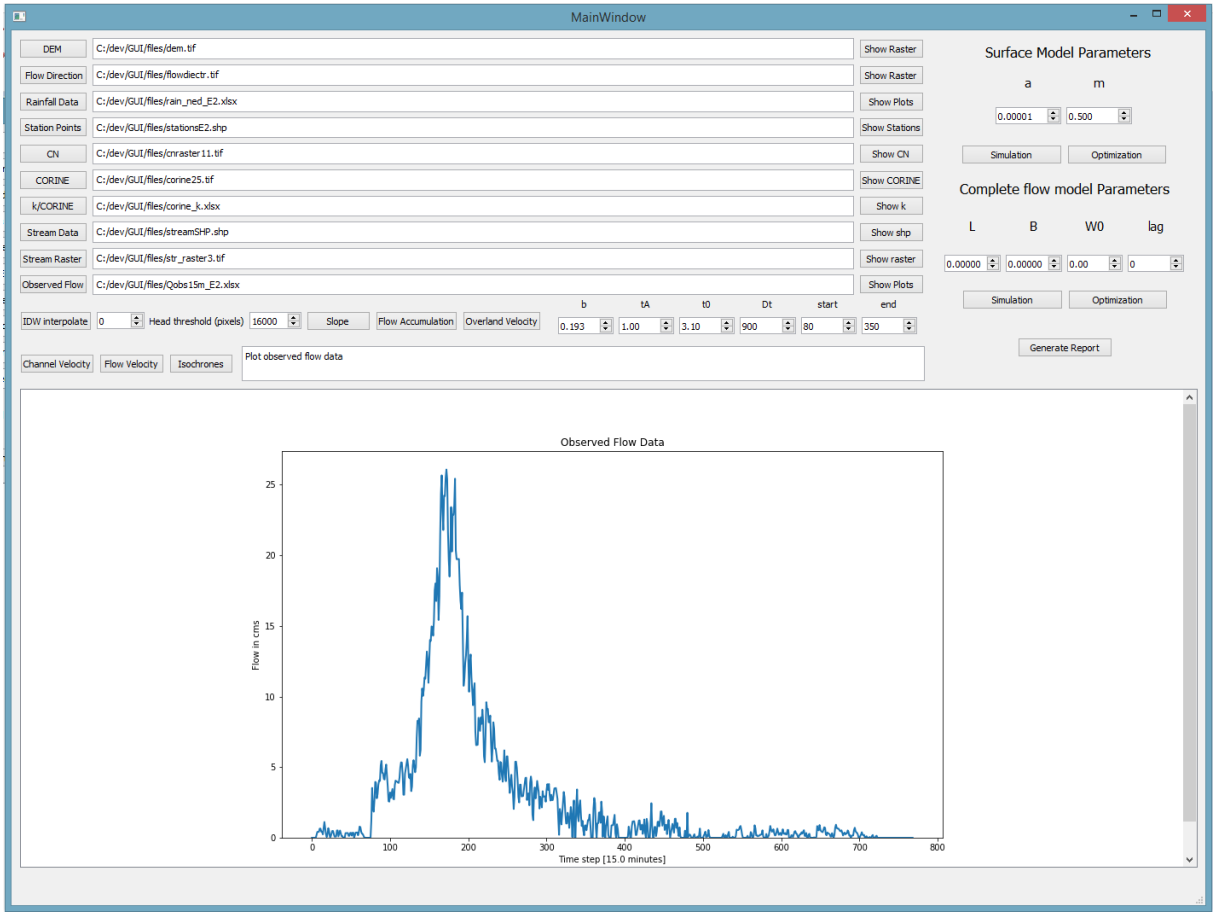


Figure 7.7: Example of observed hydrograph import.

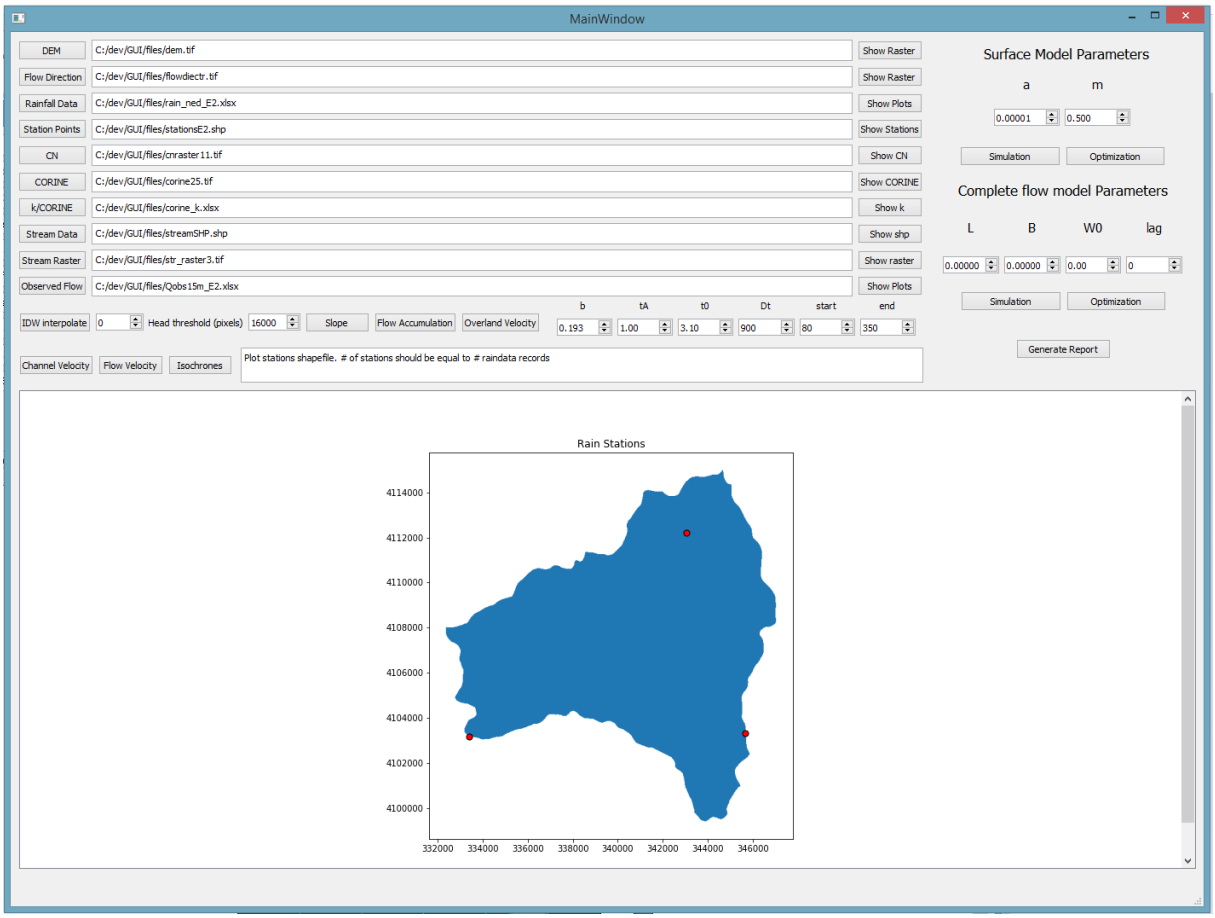


Figure 7.8: Example of stations' shapefile import visualized on top of the basin footprint.

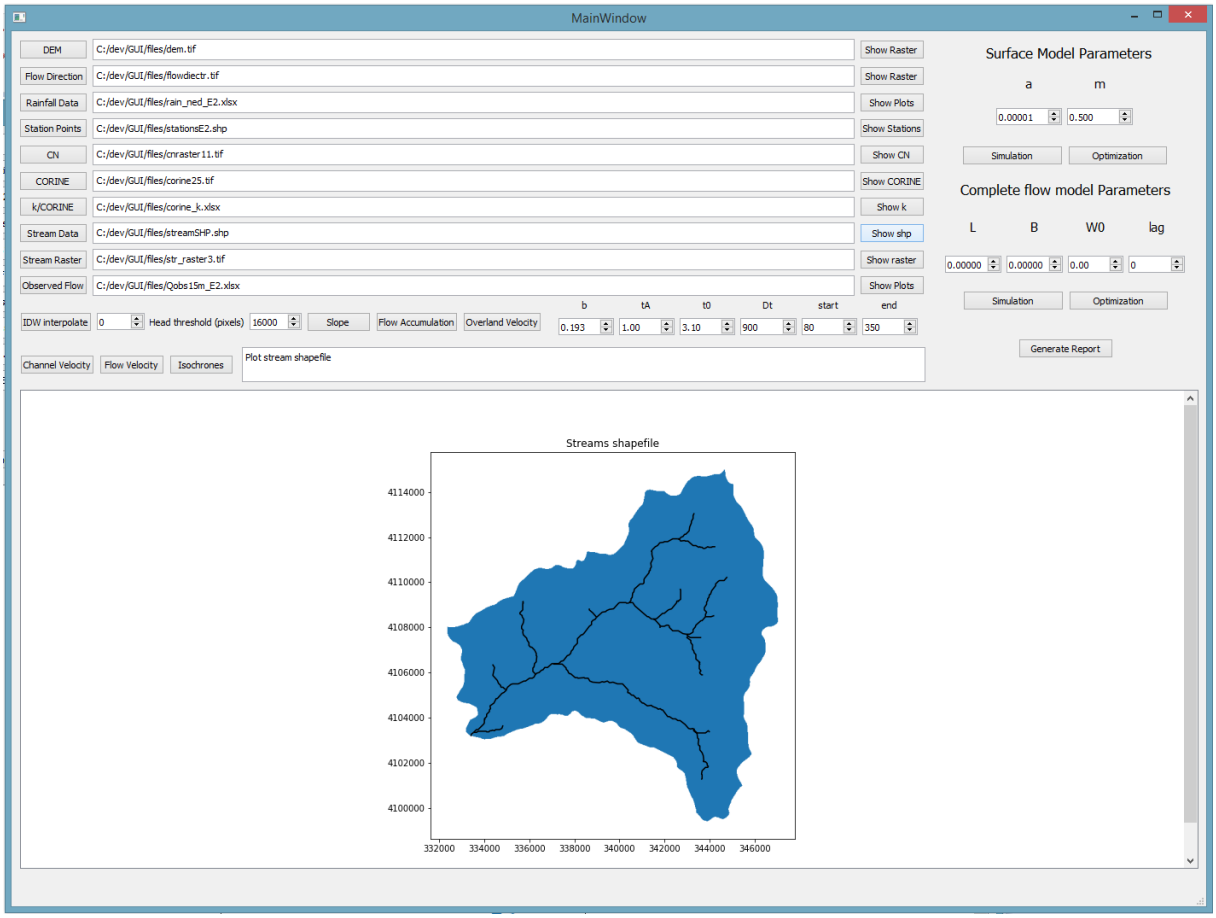


Figure 7.9: Example of the streams network shapefile import, visualized on top of the basin footprint.

Under the import wizard section there are actions available to the user for data processing. These include the IDW rainfall spatial allocation to cells, the computation of cell slope, flow accumulation and relevant velocities and visualization of the basins isochrones areas. There are input widgets (comboboxes) that enable the user-defined input of the respective parameters for the actions (e.g. the parameters for time of concentration estimation, sub-basin area threshold etc.). Examples of interaction are given in the following figures.

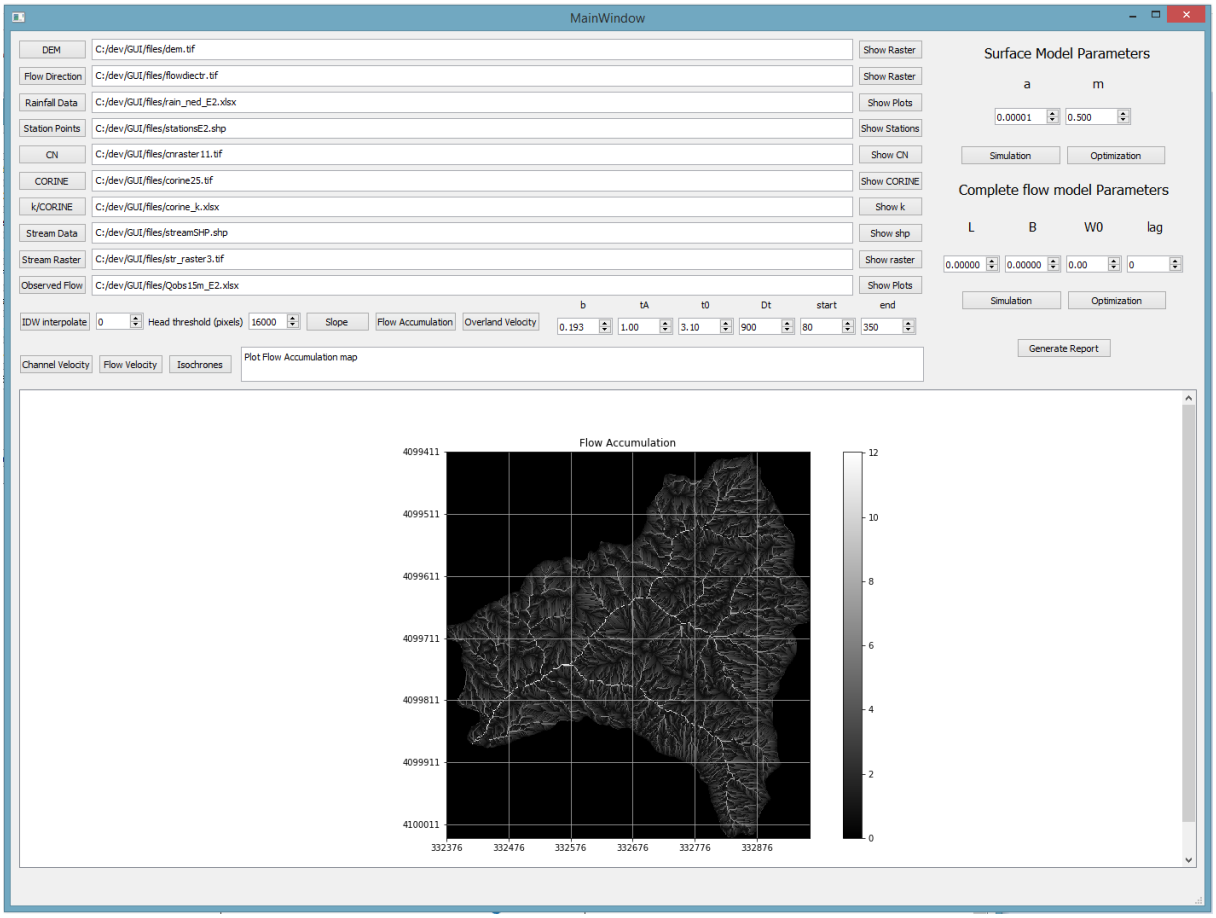


Figure 7.10: Example of flow accumulation computation.

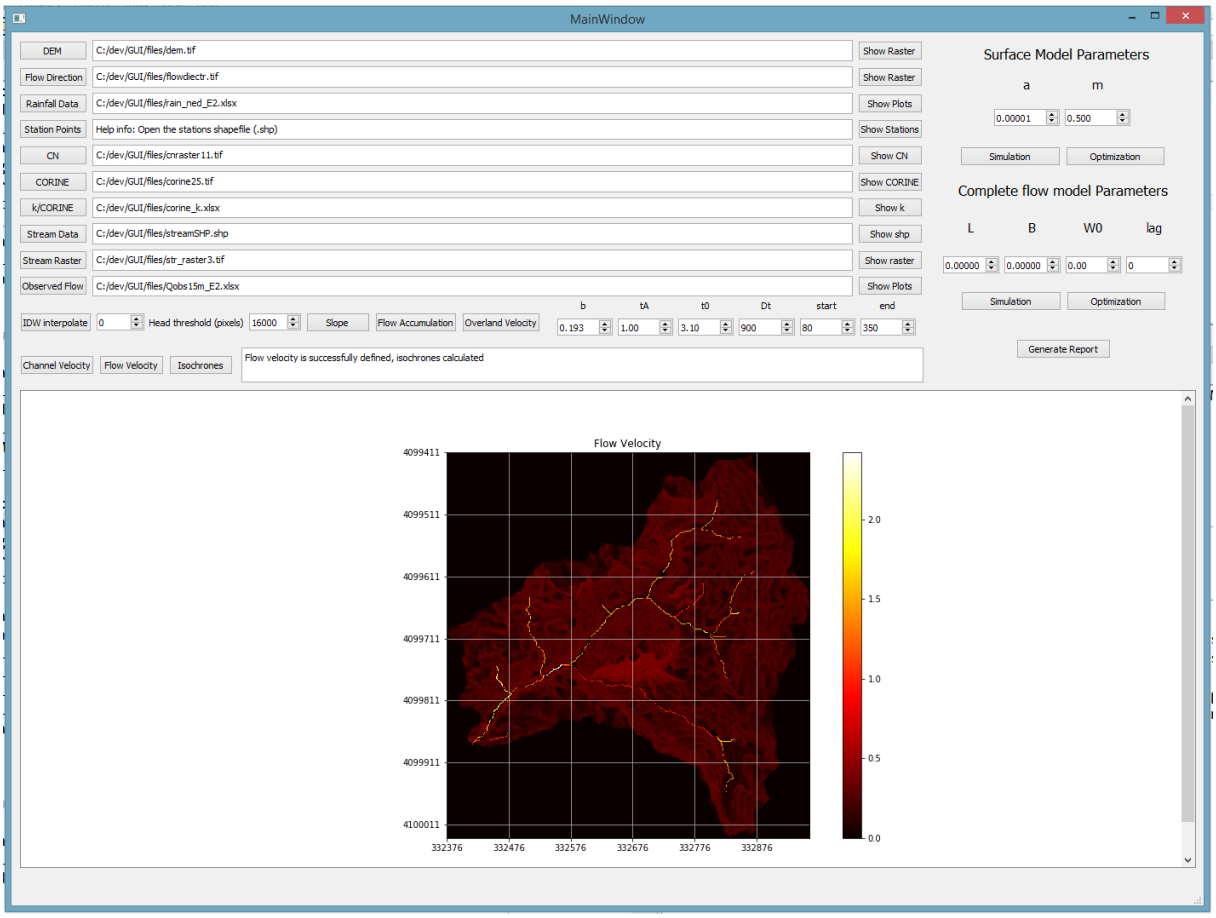


Figure 7.11: Example of calculation the final cell velocity values.

On the top right side of the window are the user defined inputs of parameters used for the simulation of an event supplying either the parameters for the surface model, or adding the parameter values needed for the complete model. The user is also able to run an optimization for either model and the parameter values will populate the input widgets at the end of the optimization. The generate report button saves results of optimization (parameter values, simulated hydrograph and metrics) in a text file. An example of visualization is given in Figure 7.12.

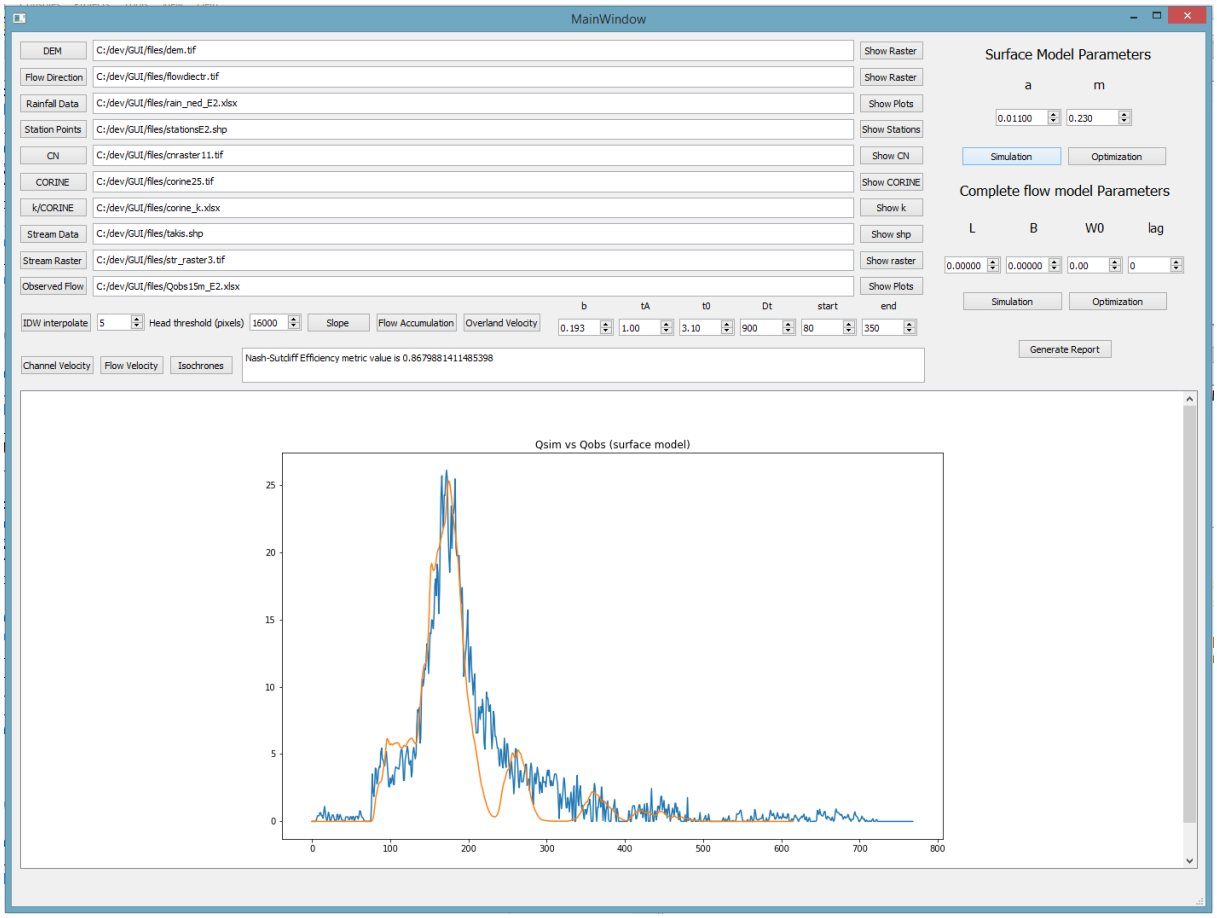


Figure 7.12: Example of simulation results.

8 Study area and data

The scope of this study is the development and documentation of a fully distributed model for event-based hydrological modelling that could be used as a stand – alone application.

Hence, the proposed scheme is not established upon a specific study basin, but is rather a general framework aimed for being easily implemented in different size basins. However, a study area is selected, in order to analyse and report the functionality and the results of the proposed model. In this chapter, we present the study basin as well and the available data.

8.1 Nedontas river basin

Nedontas river is located in the region of Western Peloponnese, Greece and belongs to the Water Department of Western Peloponnese (GR01). A general overview of the area is given in Figure 8.1. Nedontas passes through the city of Kalamata, in the prefecture of Messenia. The special feature of this site is a deep narrow gorge, with a length of 9 km, lying between Chani Lagou and the military shooting area, north of Kalamata. Nedontas river springs from the western slopes of Taygetos, and flows into the Messenian gulf, west of the harbor of Kalamata, with a total length of 26 km. In the area exists a network of meteorological and hydrometric stations. The cross section of the basin was selected upstream of the urban area, and in particular at the Nedontas hydrometric station, at the Bakas's Quarry, where the cross section is upgraded downstream.

This watershed is suitable for our study, as there don't exist structures that could affect its hydrological regime (e.g. dams, deflections, reservoirs). Due to the karstification, a high percentage of runoff water in the Nedontas riverbed infiltrates through the limestone, thus contributing to the enrichment of the groundwater and maintaining the relatively low runoff towards the river mouth. The geomorphological development of the Nedontas gorge is due to extensive erosion which occurred during the post alpine elevation of the area; erosion was promoted across large NE-SW striking faults.

A further factor contributing to the choice of this particular case study, is the characteristics of the overland flow, which is the main component of discharge. It occurs through the well-formed hydrographic network with relatively straight sections at least at the locations of the existing hydrometric stations. The meteorological data collected from a station near the airport (6 km west of Kalamata) provide information on the area's water potential.

In the south of Prefecture of Messinia (Finikounda – Methoni), the rainfall is in average 600 mm, 1500 mm in the mountainous areas and 800-1200 mm in central, northern plain and semi - mountainous areas.

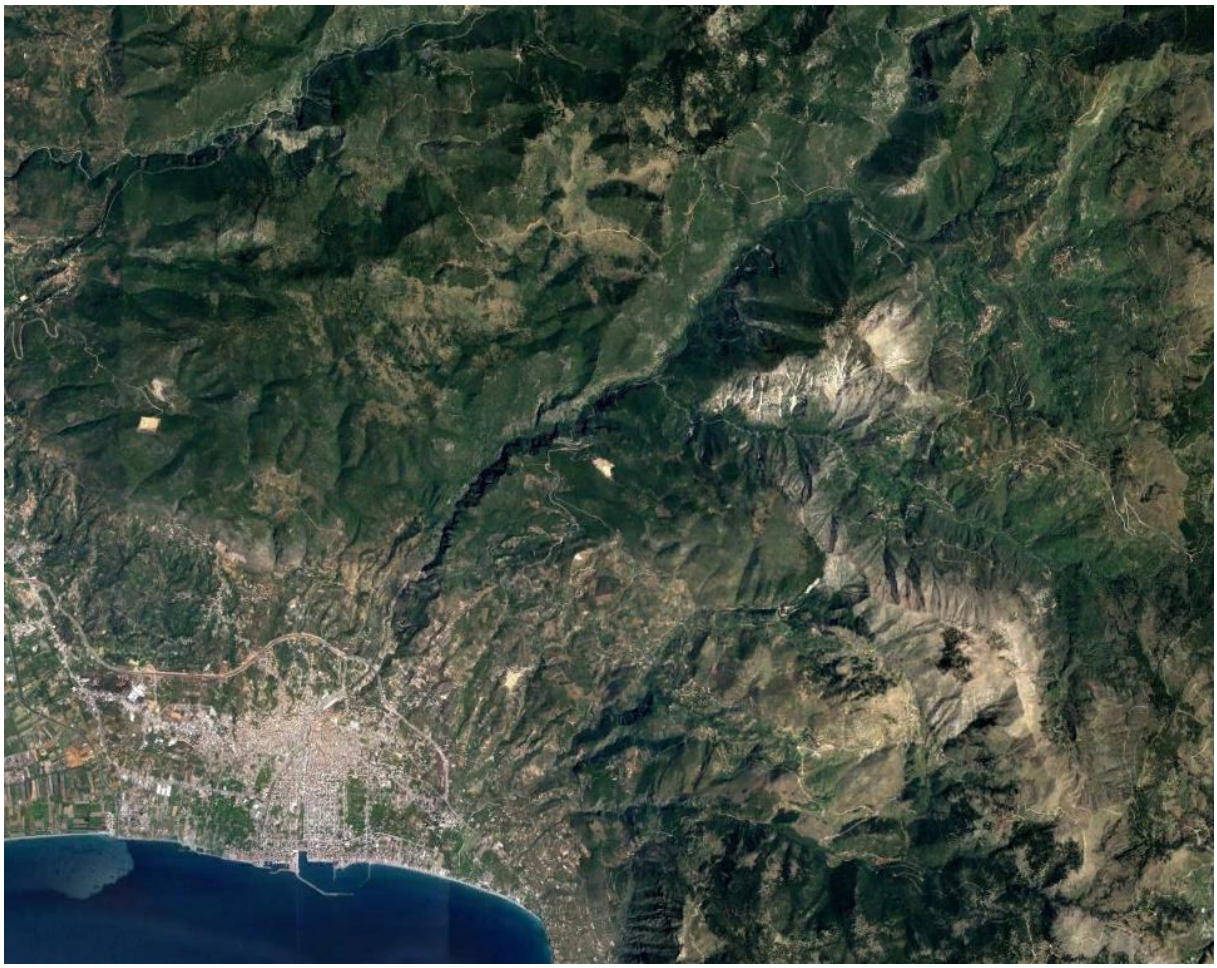


Figure 8.1: Satellite imagery of the general area around Nedontas river, as seen in Google Earth Pro.

8.2 Processing of the Geospatial Information

The available geospatial data used for the examined catchment area are altitude information, land use cover, geological structures as well as the locations of the metric stations. Figure 8.2 shows the Digital Elevation Model (DEM) with a spatial resolution of 25 m per pixel. Total basin area is equal to 119.3 km^2. As it can be observed from the altitude histogram (Figure 8.3) of the examined data, the minimum altitude value is approximately 93 m and the maximum 1 715 m.

In Nedontas basin the slopes are generally steep, as 75% of the cells have slope greater than 18%. In the northern areas in particular, slope exceeds 100% (Figure 8.4).

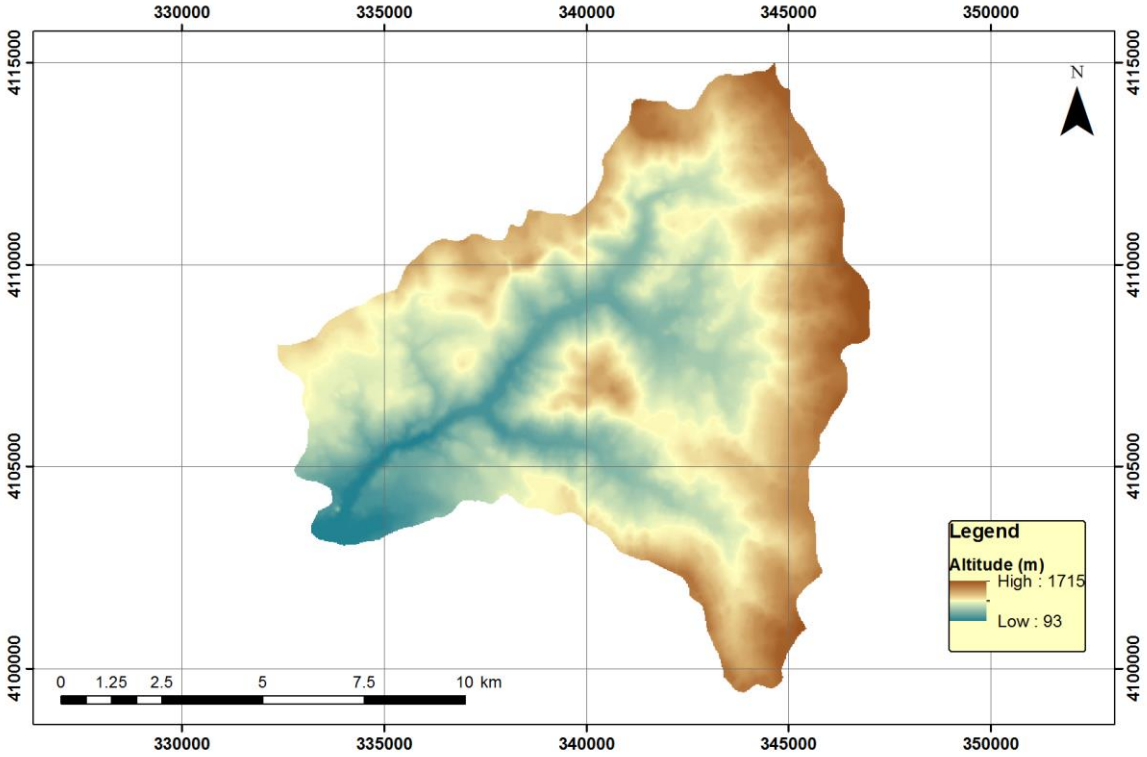


Figure 8.2: DEM of the basin under study.

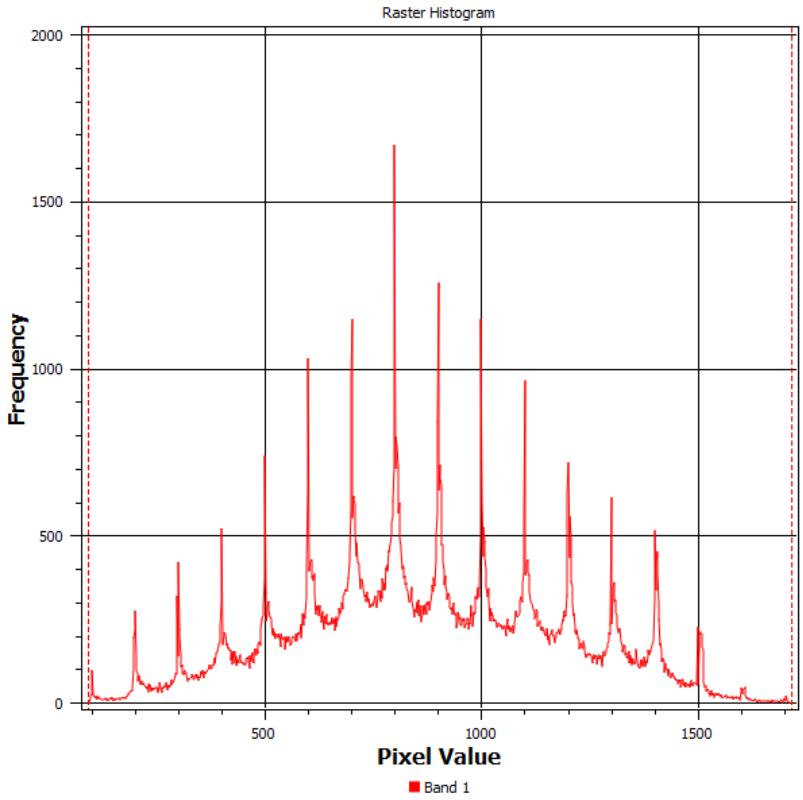


Figure 8.3: Altitude histogram of the basin.

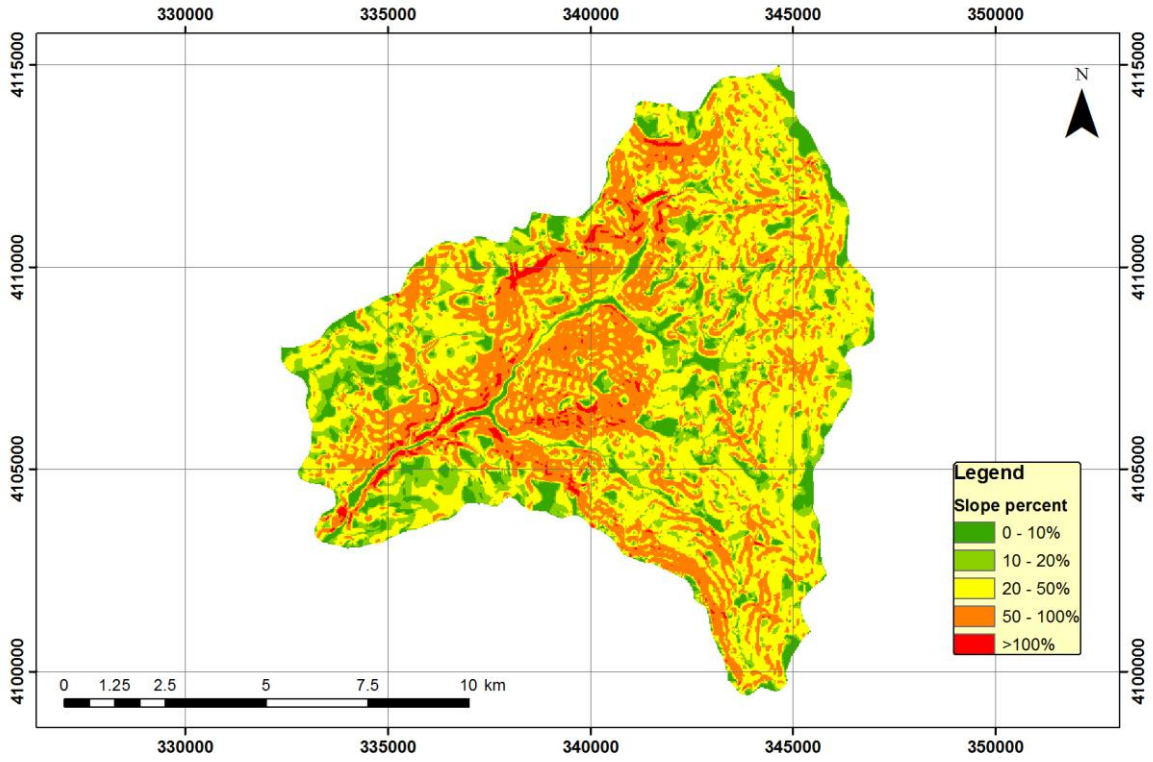


Figure 8.4: Slope classification.

8.2.1 Hydrographic network

The hydrographic network consists of the main branch of Nedontas and three major tributaries, i.e. Nedousa, Alagonia and Karveliotis. The first to form the upper reaches of the examined river, while the third runs across the southwest part of the basin. Figure 8.5 illustrates the hydrographic network of the basin. The main water body of the area collects the smaller streams and creeks from the upstreams and reaches the outlet of the basin, which is located in the Messenian Gulf. Antoniadi (2016) estimated the unit time of concentration, t_0, equal to 3.1 hours and shape parameter, β equal to 0.193.

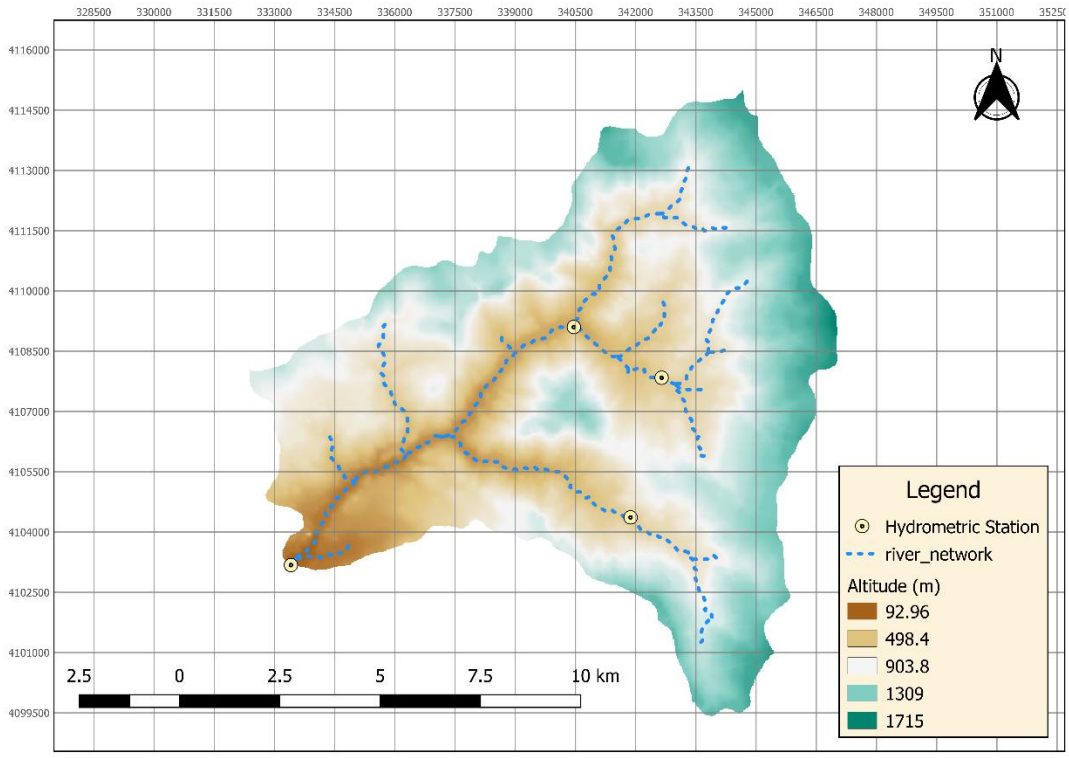


Figure 8.5: Hydrological network in Nedontas basin.

8.2.2 Land use

The digital land use model derived from Corine Land Cover 2012. Figure 8.6 depicts the land uses for the studied basin. It is obvious that the main land use is broad – leaved forest and coniferous forest.

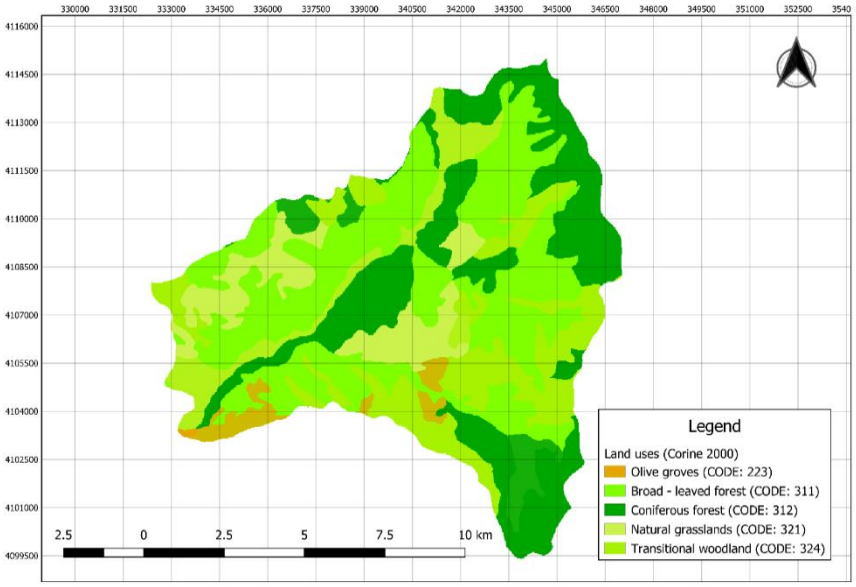


Figure 8.6: Corine Land Cover Classification.

8.2.3 Geological background

As seen in Figure 8.7, most of the central part of the basin is comprised of the Tripolis Zone which is karstified. The east side of the basin is comprised of quartz geological formations.

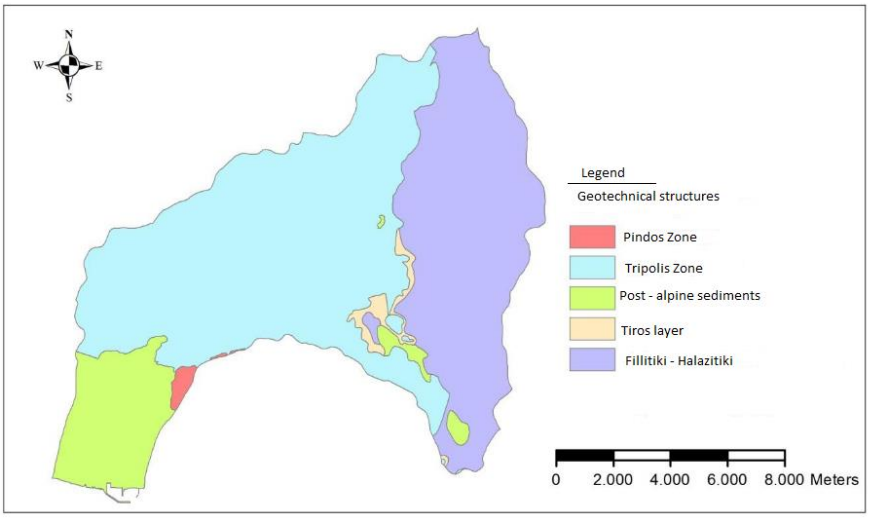


Figure 8.7: Geological background of Nedontas basin.

8.2.4 Permeability

The classification of the soils infiltration rate according to SCS is high (Class A), medium to low (categories B and C) and at a very low rate (Category D). Due to the small size of categories B and C, they were consolidated. The following figure depicts the water permeability.

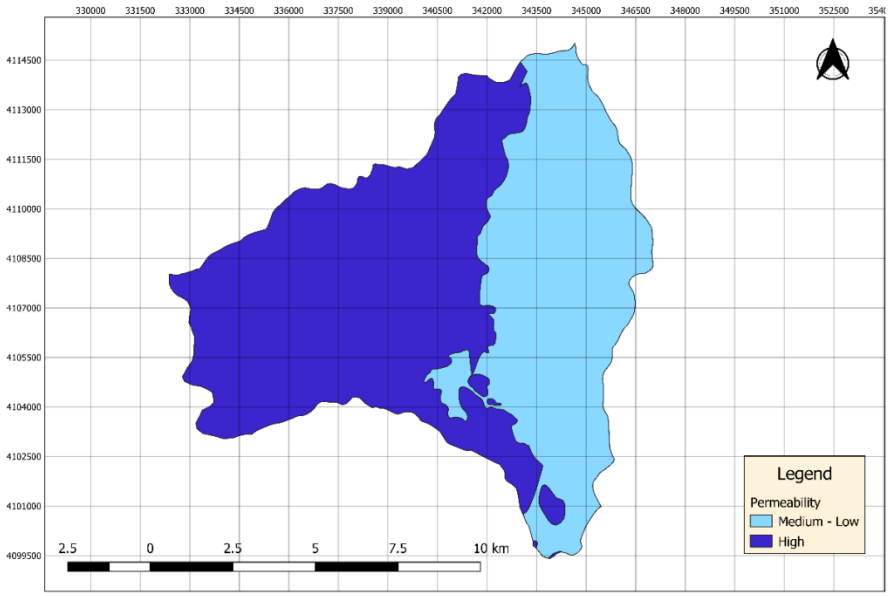


Figure 8.8: Permeability map of Nedontas basin.

8.2.5 Estimation of CN values for AMC II conditions

CN values for AMC II conditions are calculated according to the methodology described in 5.3.4 and are presented in Figure 8.9. The mean value of CN is 62.35.

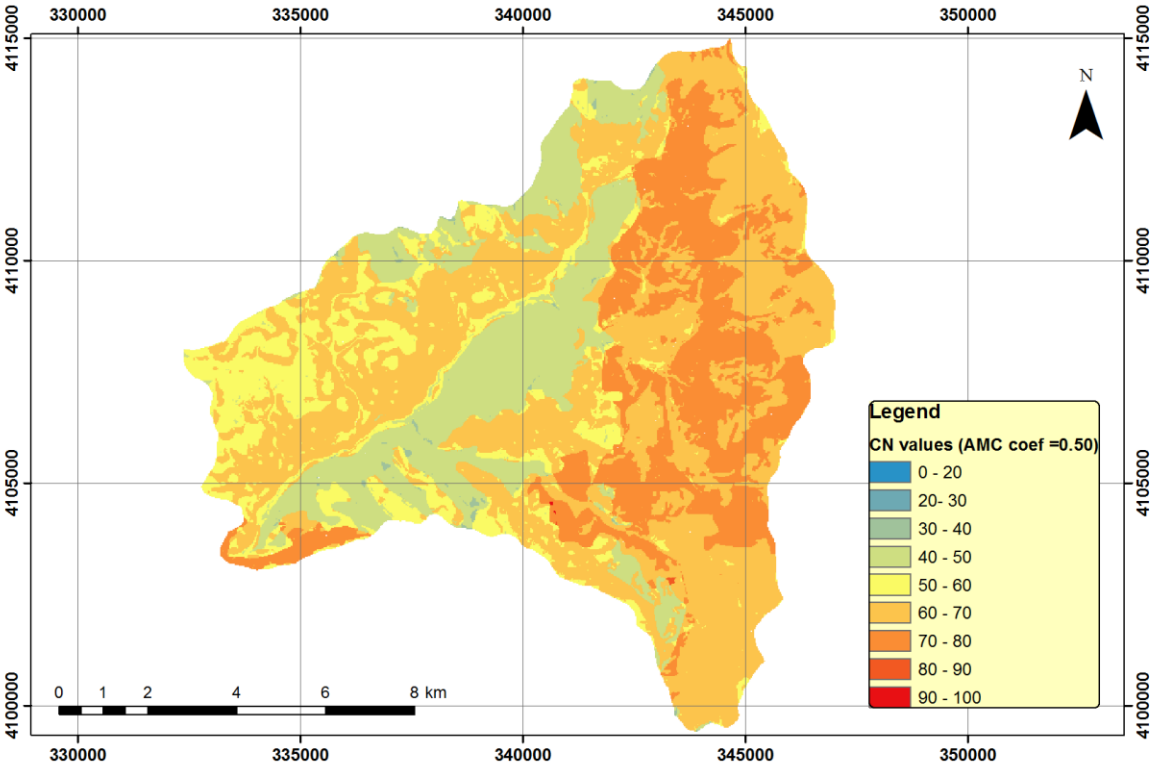


Figure 8.9: CN values for AMC II conditions.

8.2.6 Flow direction and flow accumulation

In Figure 8.10 the flow direction map is presented. In Figure 8.11 the flow accumulation of the examined catchment is illustrated, with a threshold of sub-basins equal to 2.5 km^2.

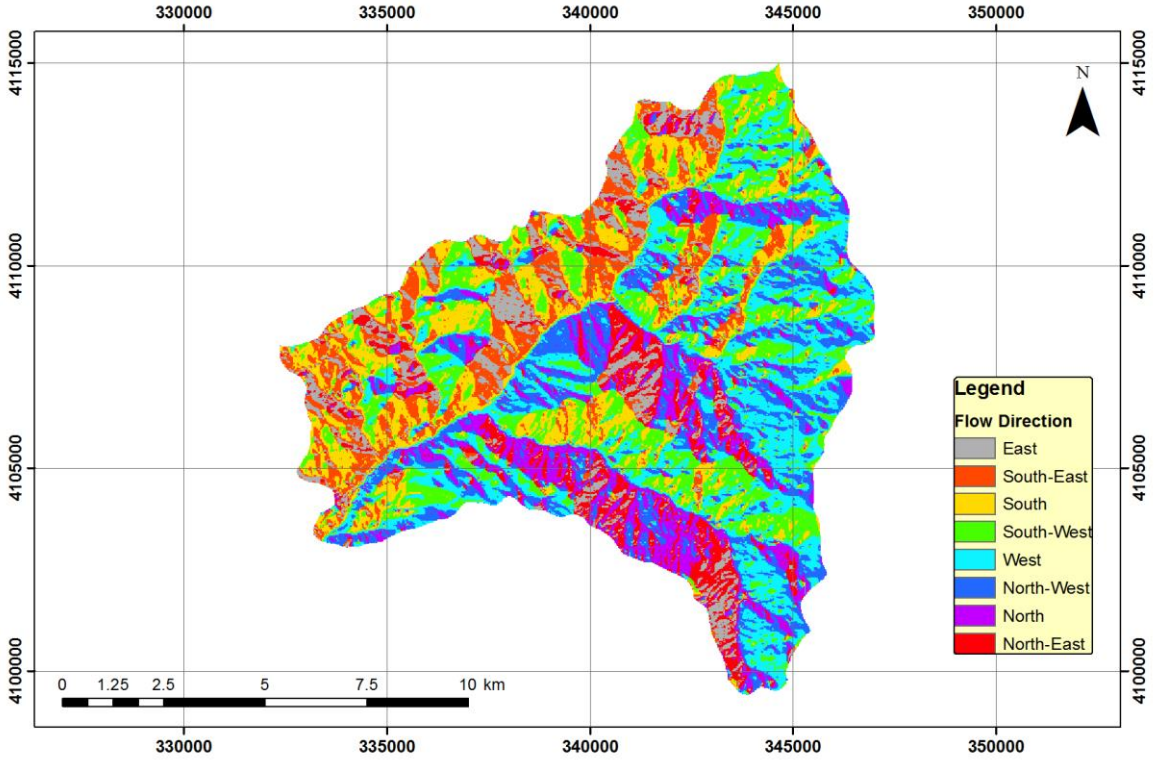


Figure 8.10: Flow direction raster.

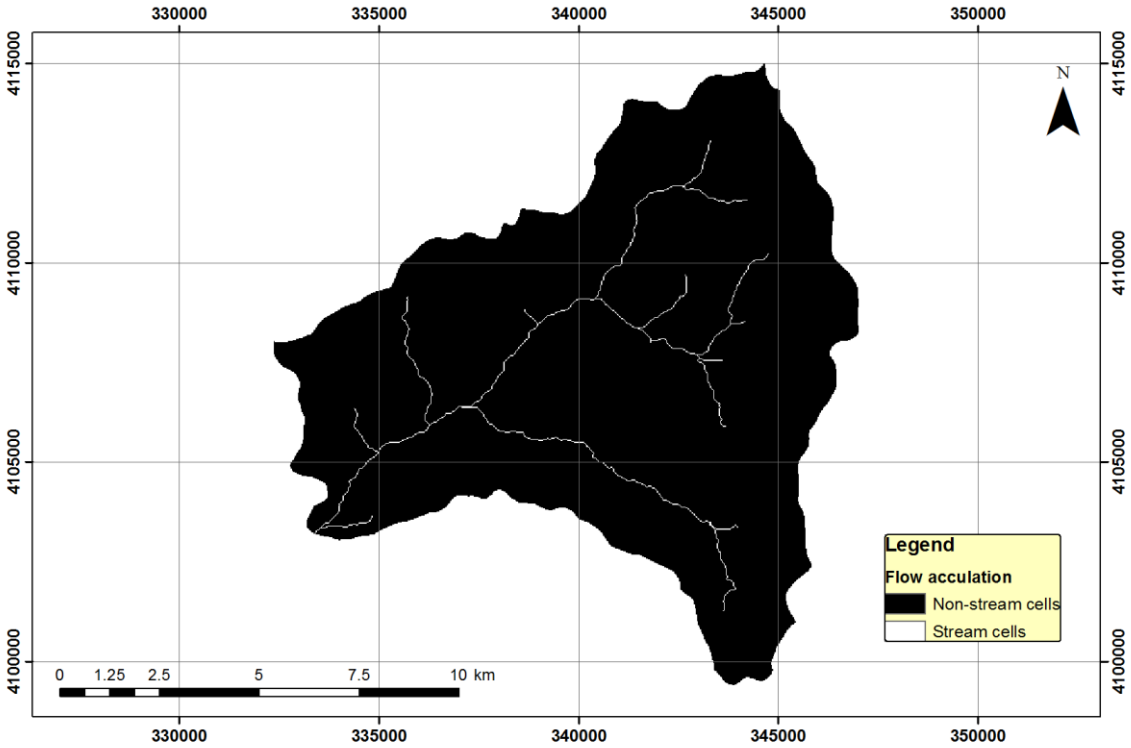


Figure 8.11: Flow accumulation with threshold of 2.5 km^2.

8.2.7 Analysis of the stream network

Using a sub-basin threshold of 2.5 km^2 and the common GIS hydrology tools the water network of Figure 8.12 emerges. Streams are named by ID values. The attributes *slope,length* and *Manning* coefficient are presented in Table 8.1. Manning coefficients are estimated macroscopically by means of satellite imagery interpretation.

Table 8.1: Stream network attributes

ID	Length (m)	Slope %	Manning coeff.
1	1483.31	0.135589	0.07
2	1889.64	0.119282	0.07
3	4150.66	0.068437	0.05
4	2316.43	0.110869	0.07
5	396.309	0.193031	0.07
6	523.51	0.166129	0.07
7	1704.03	0.024108	0.03
8	1342.08	0.060317	0.05
9	2094.73	0.057592	0.07
10	448.144	0.0459	0.05
11	87.5	0.052914	0.07
12	1342.92	0.076423	0.05

13	1251.79	0.081723	0.07
14	106.066	0.09	0.07
15	612.5	0.097796	0.07
16	3059.25	0.025264	0.03
17	3821.84	0.149473	0.07
18	995.376	0.068587	0.03
19	2156.43	0.092162	0.07
20	1443.49	0.273469	0.07
21	1532.63	0.016188	0.03
22	7664.86	0.054223	0.07
23	2616.26	0.040413	0.03
24	1488.46	0.068998	0.07
25	870.789	0.162783	0.07
26	216.001	0.021713	0.03
27	2756.11	0.127992	0.07

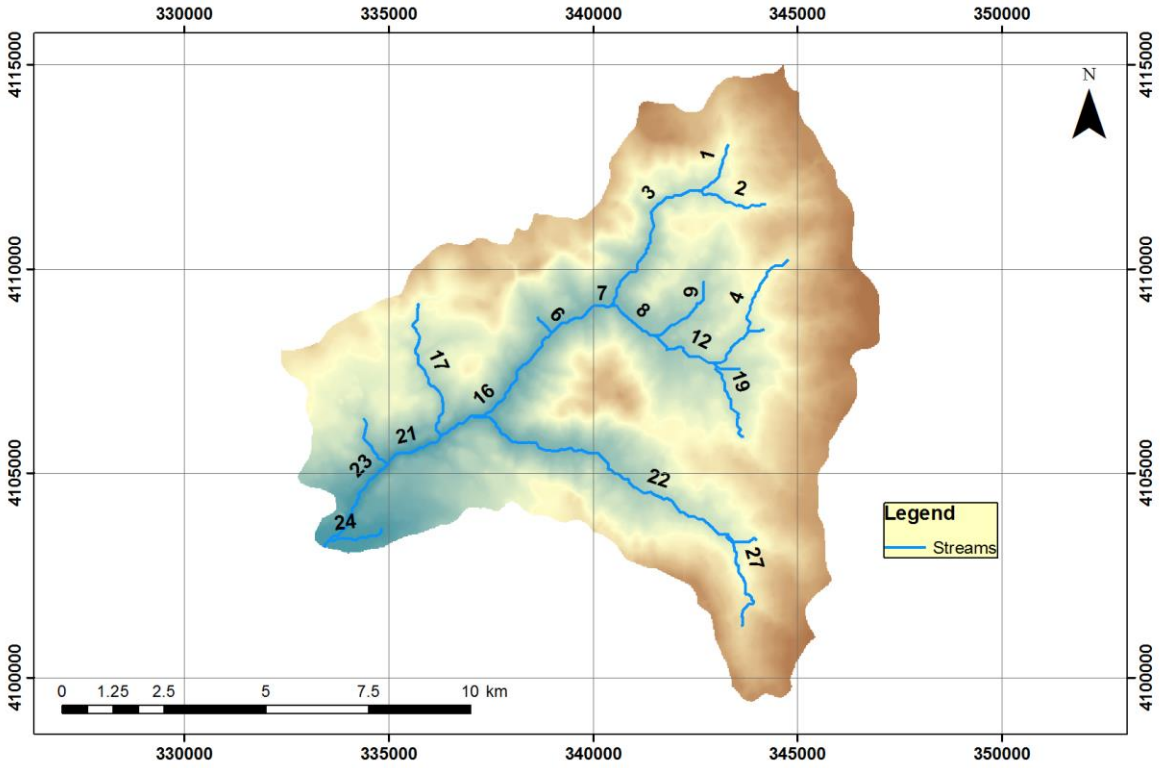


Figure 8.12: Streams by ID.

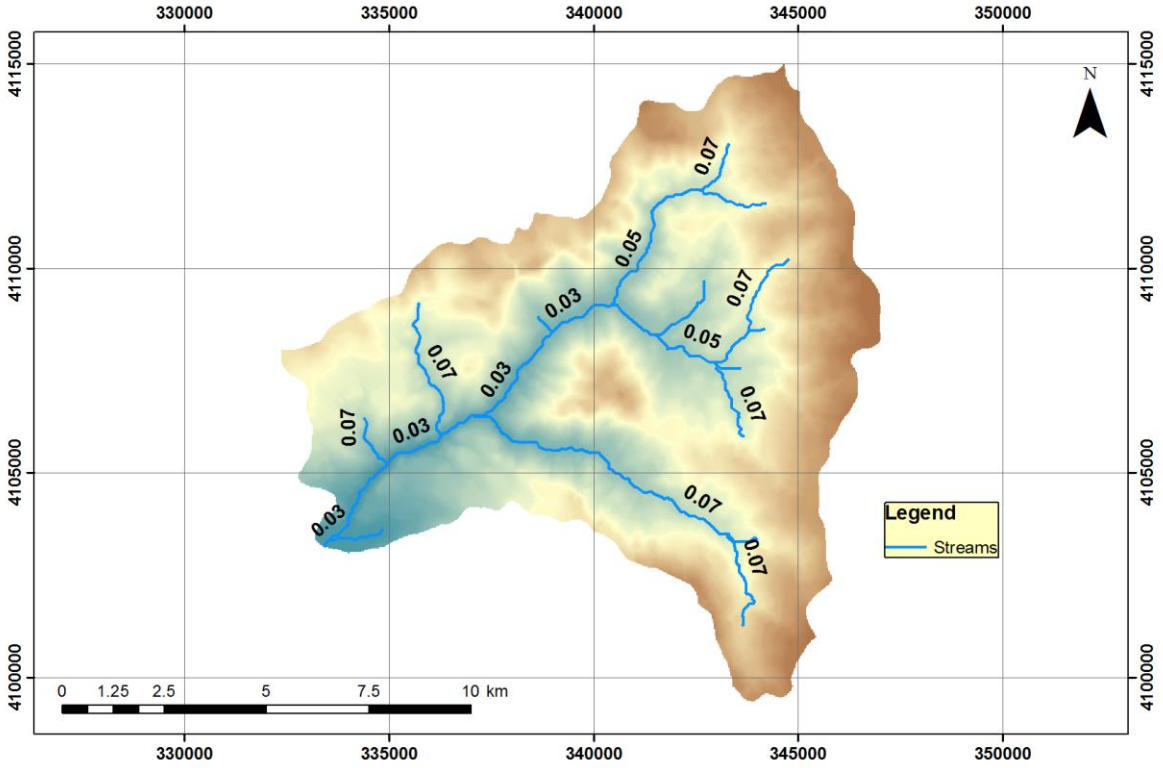


Figure 8.13: Manning values of the stream segments.

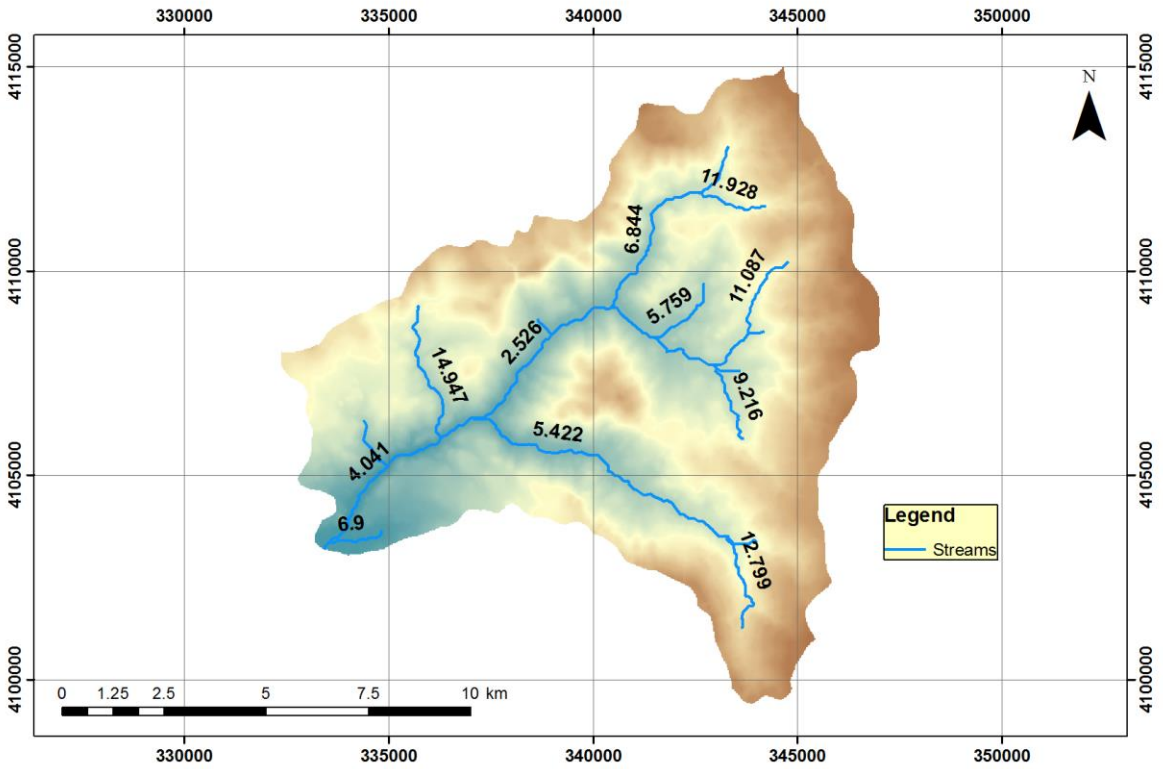


Figure 8.14: Stream segments average percent (%) slope.

8.3 Input data (rainfall and observed hydrographs)

The events that are used for the implementation of the models are presented. In this study, two events with a 15-minute observation interval are used in the basin of Nedontas.

8.3.1 Event of January 16th 2013 (Event A hereafter)

This event started in 16/1/13 and ended in 19/1/13. The following gauges are used:

- Karveliotis
- Taygetos
- Nedousa
- Alagonia
- Poliani
- Kalamata – Nisaki

Figures 8.13 to 8.18 illustrate the 1st rainfall event for each of the six stations in the region of Nedontas. Figure 8.27 shows the total rainfall of the event (IDW interpolation) while Figure 8.22 depicts the mean rainfall intensity during Event B.

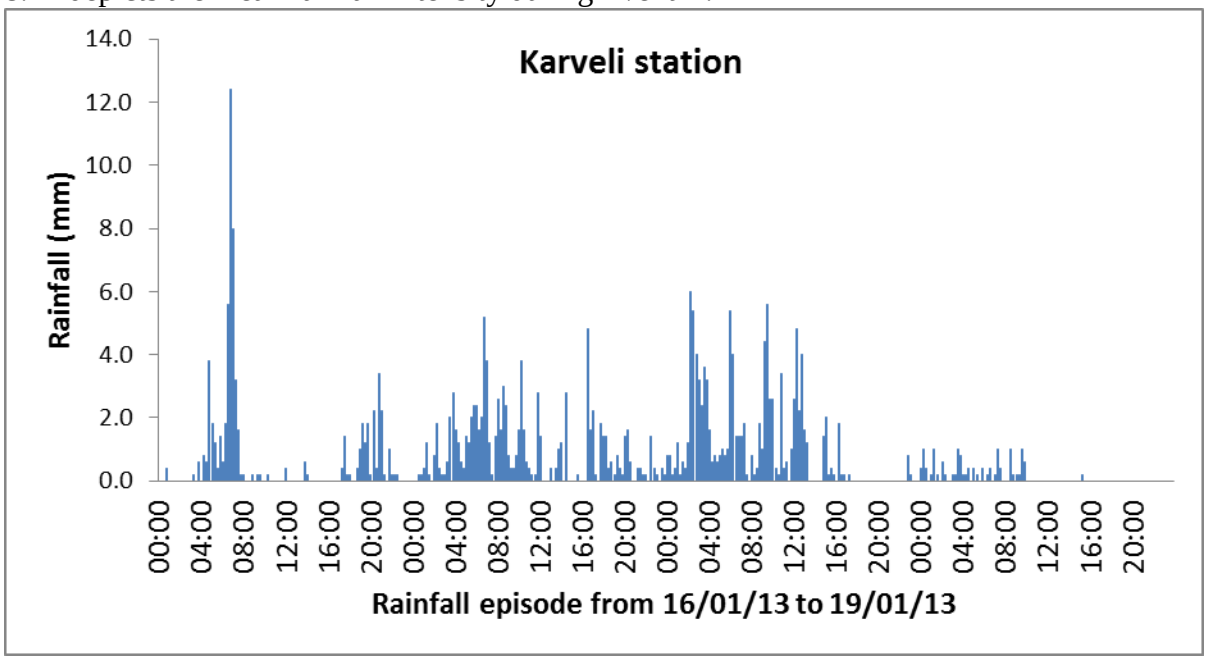


Figure 8.15: Hyetograph at Karvel rain gauge.

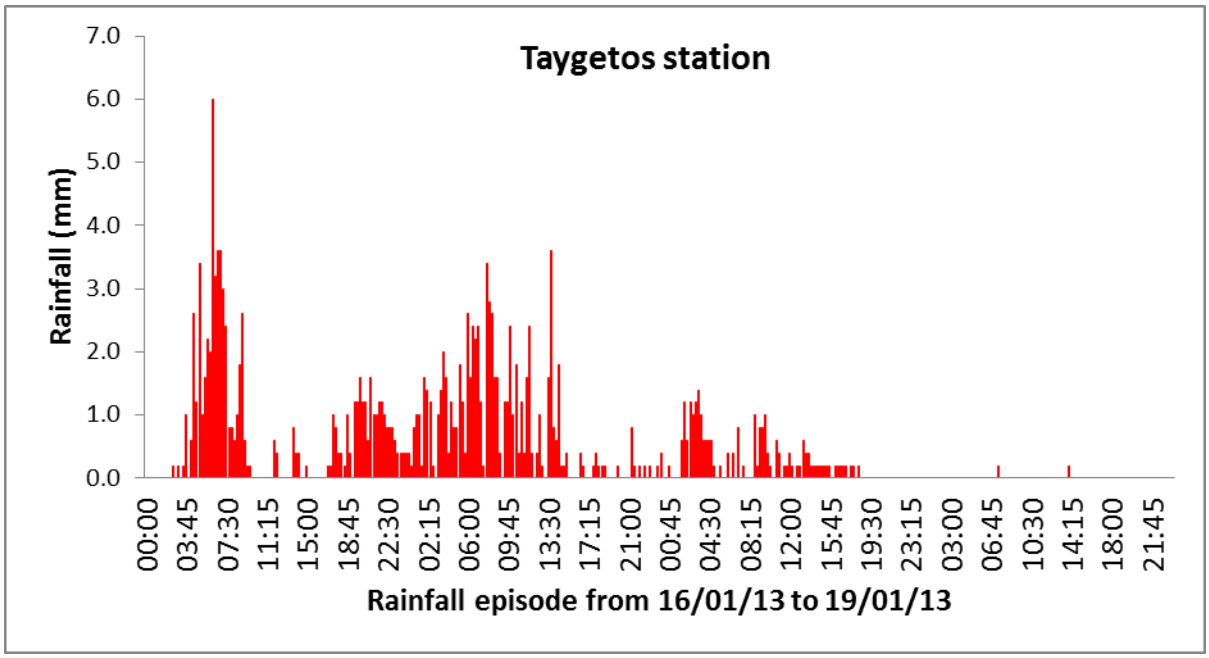


Figure 8.16: Hyetograph at Taygetos rain gauge.

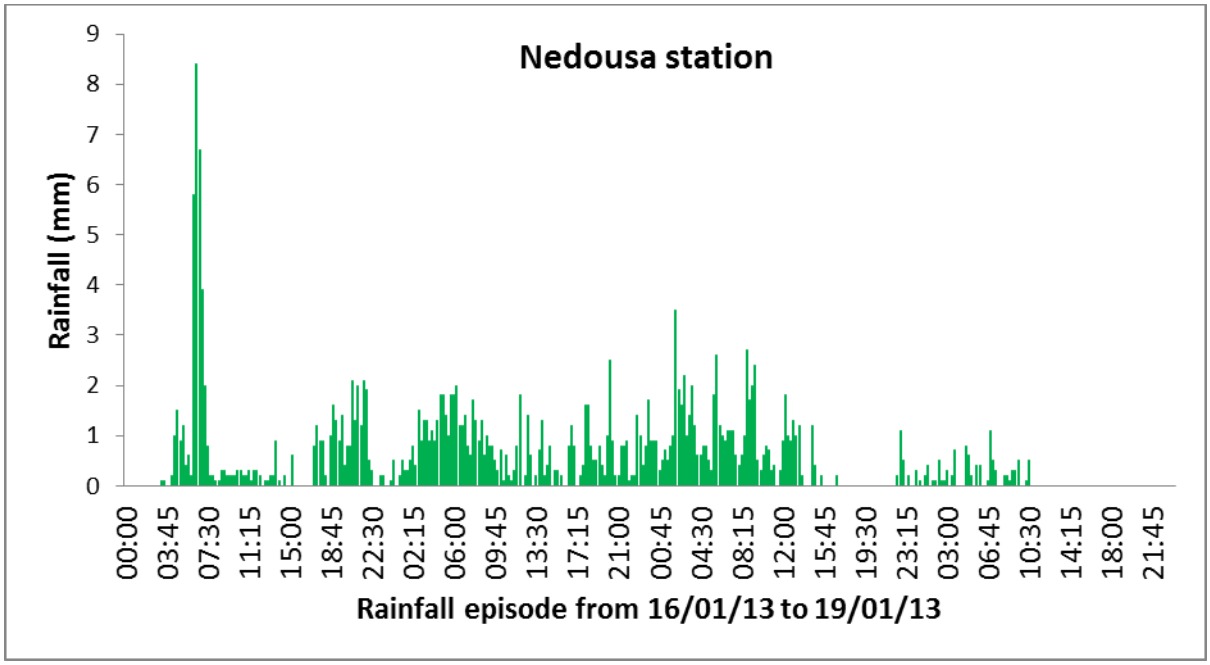


Figure 8.17: Hyetograph at Nedousa rain gauge.

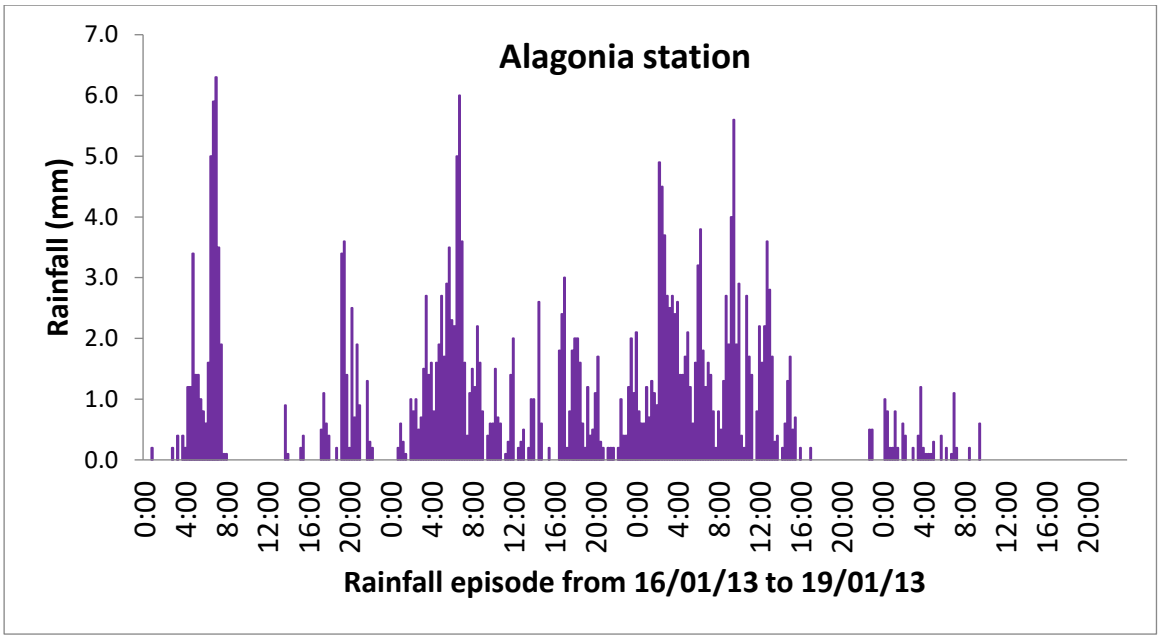


Figure 8.18: Hyetograph at Alagonia rain gauge.

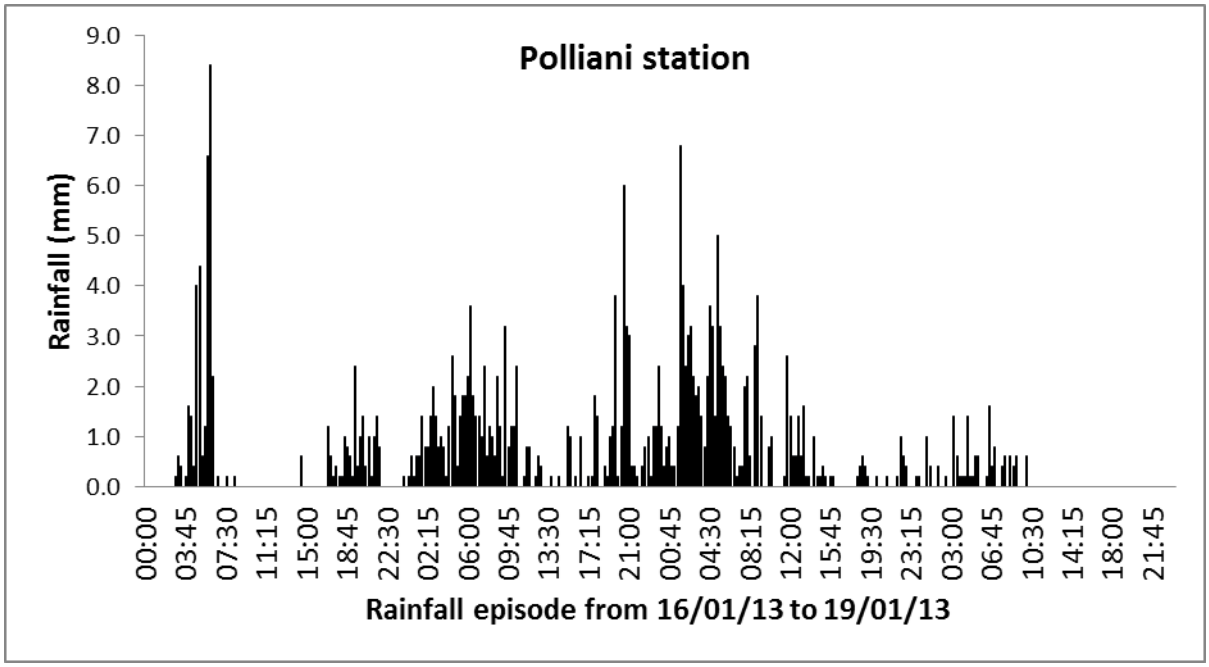


Figure 8.19: Hyetograph at Polliani rain gauge.

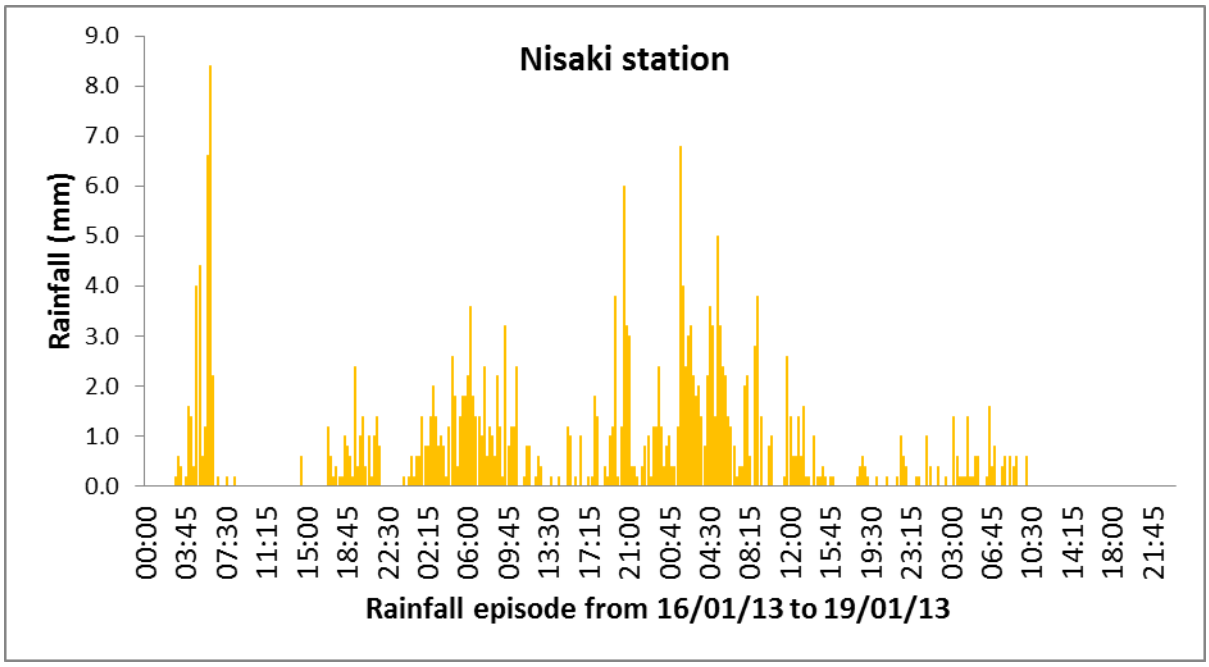


Figure 8.20: Hyetograph at Nisaki rain gauge.

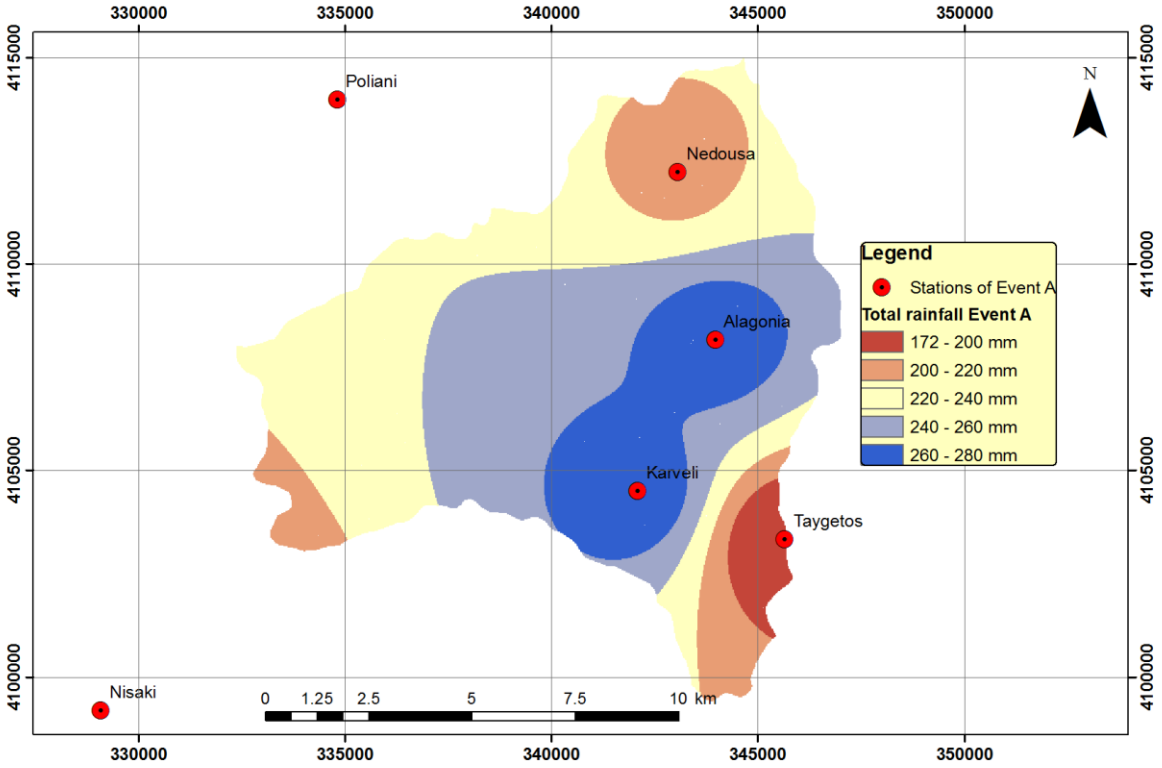


Figure 8.21: Total rainfall in mm of Event A.

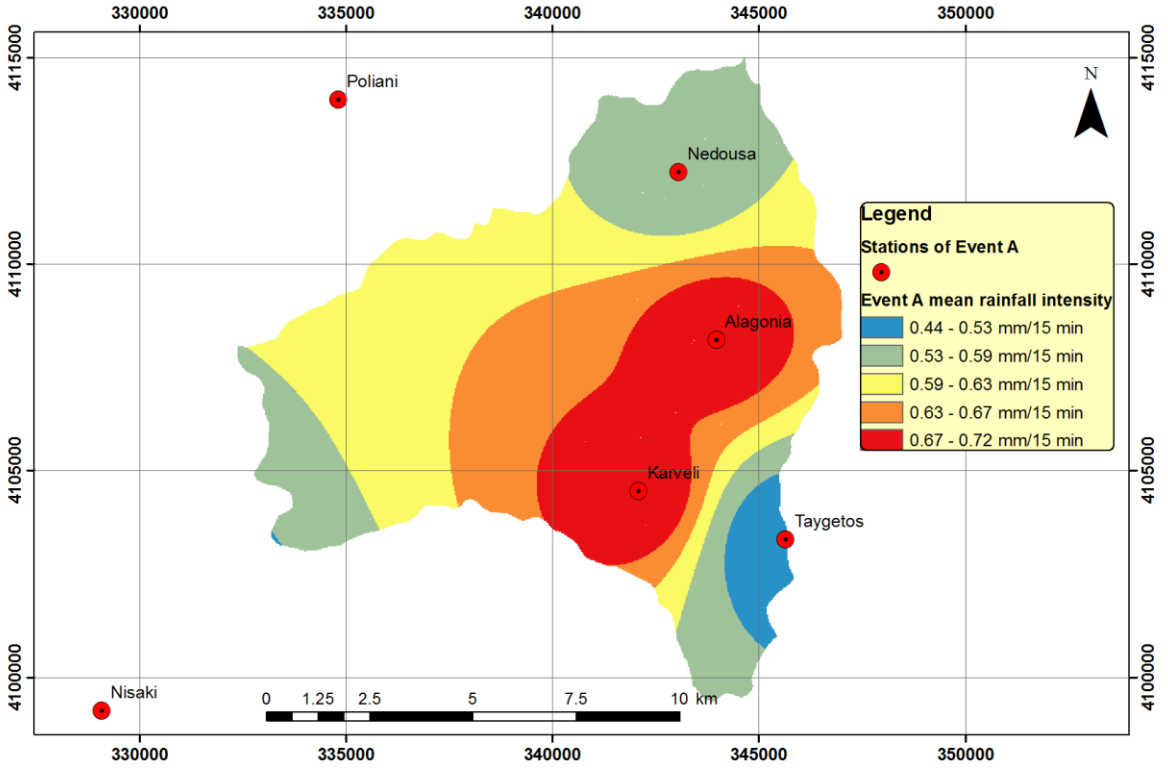


Figure 8.22: Mean intensity of rainfall, Event A.

The observed discharge for the 1st event measured in the Bakas Quarry gauge is illustrated in the Figure 8.23. Runoff values up to 21/01/13 are presented.

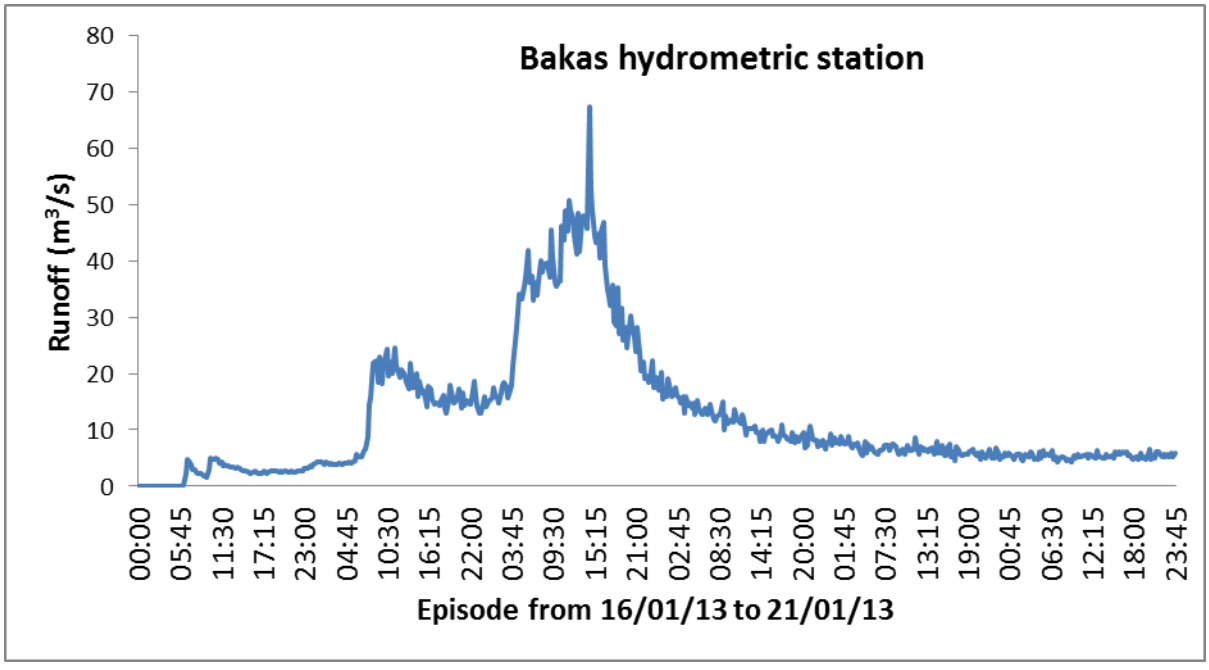


Figure 8.23: Observed discharge at Bakas Quarry gauge for Event A.

8.3.2 Event of February 6th 2012 (Event B hereafter)

The second event started in 6/2/12 and ended in 10/2/12. The following gauges were operational:

- Bakas Quarry

- Taygetos
- Nedousa

Figures 8.22 to 8.24 show the rainfall data of the above stations. Figure 8.27 show the total rainfall of the event (IDW interpolation)while Figure 8.28 depicts the mean rainfall intensity during Event B.

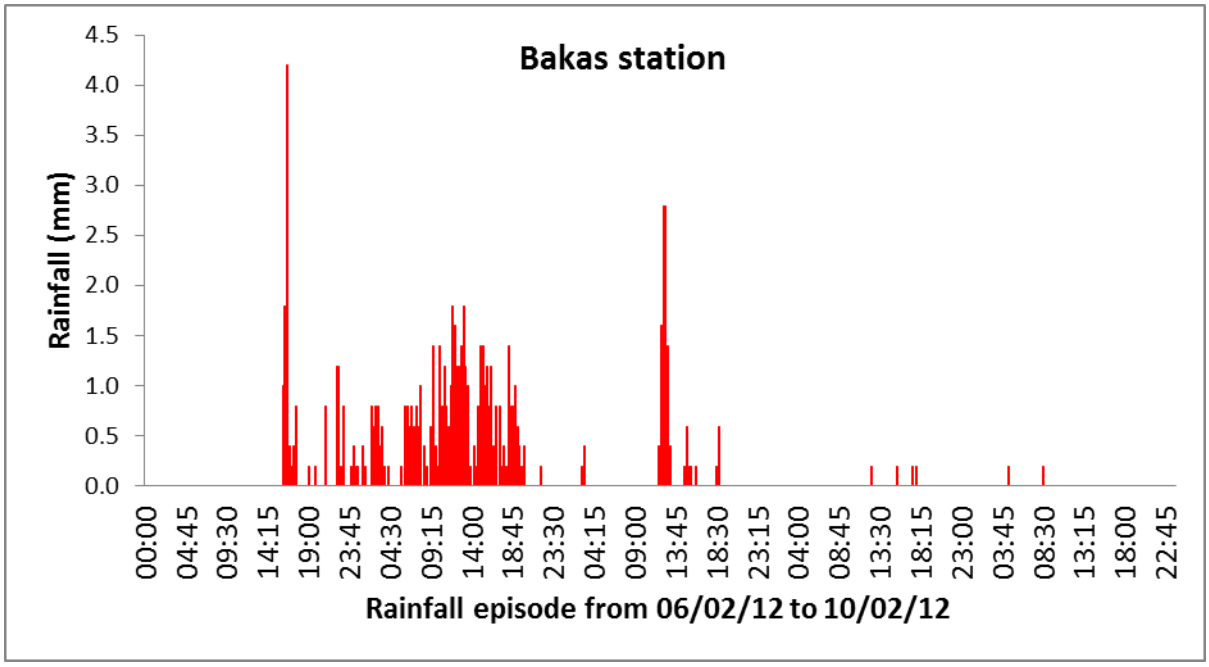


Figure 8.24: Hyetograph at Bakas quarry rain gauge.

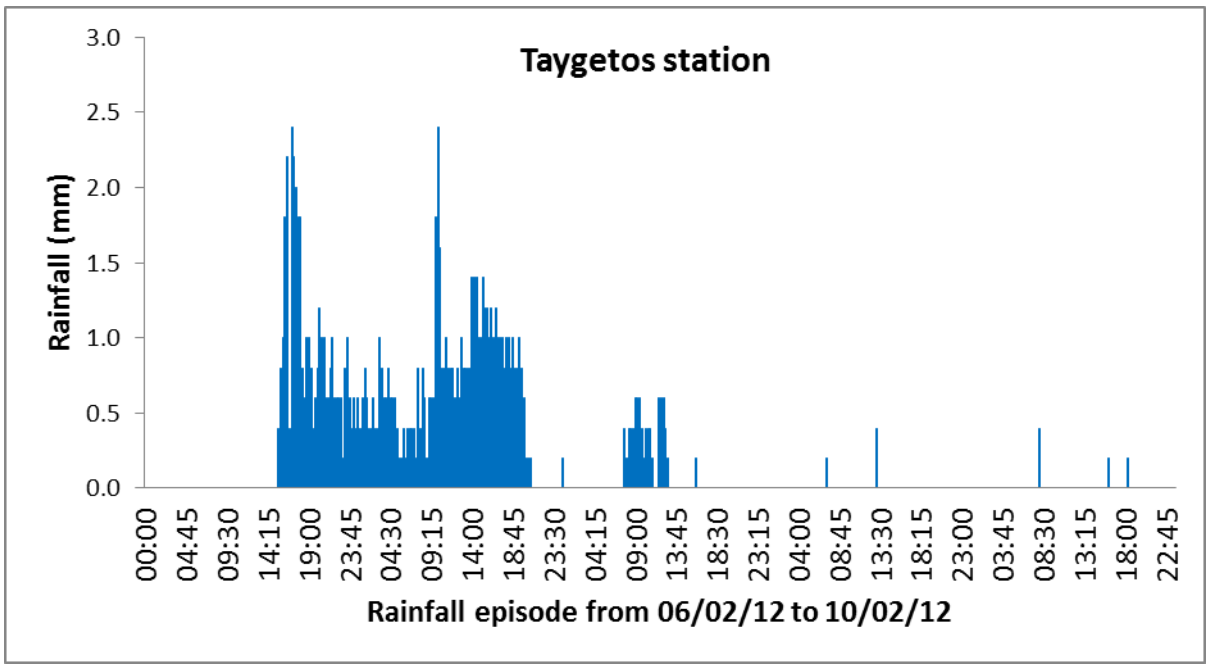


Figure 8.25: Hyetograph at Taygetos rain gauge.

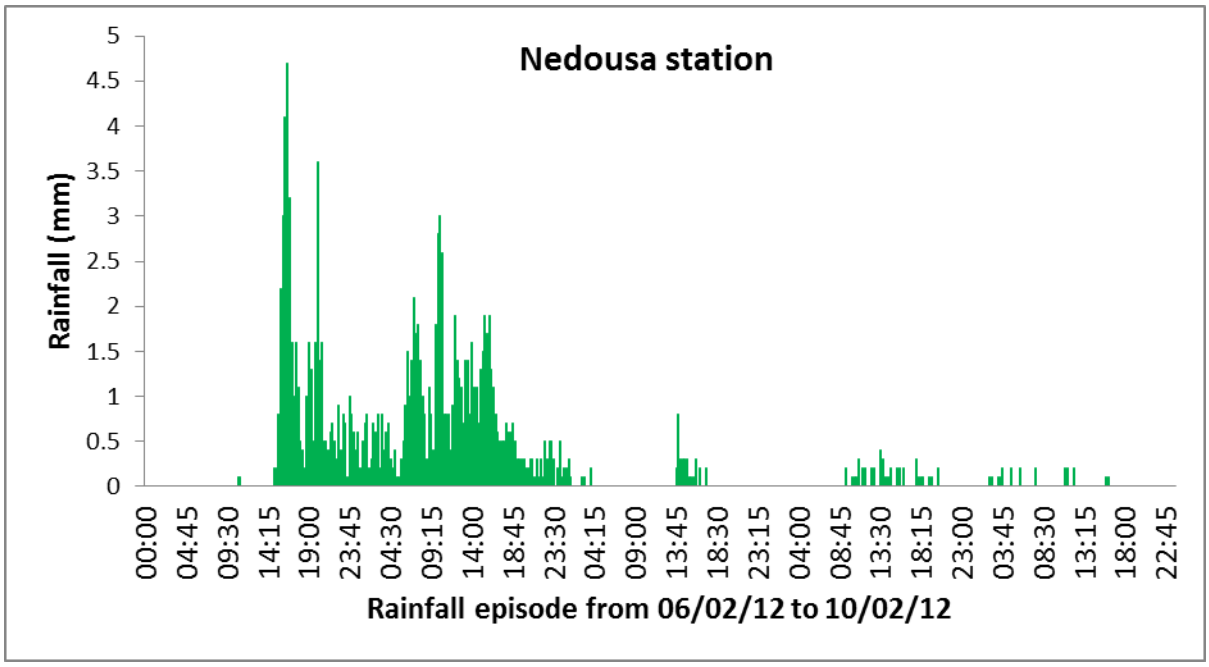


Figure 8.26: Hyetograph at Nedousa rain gauge.

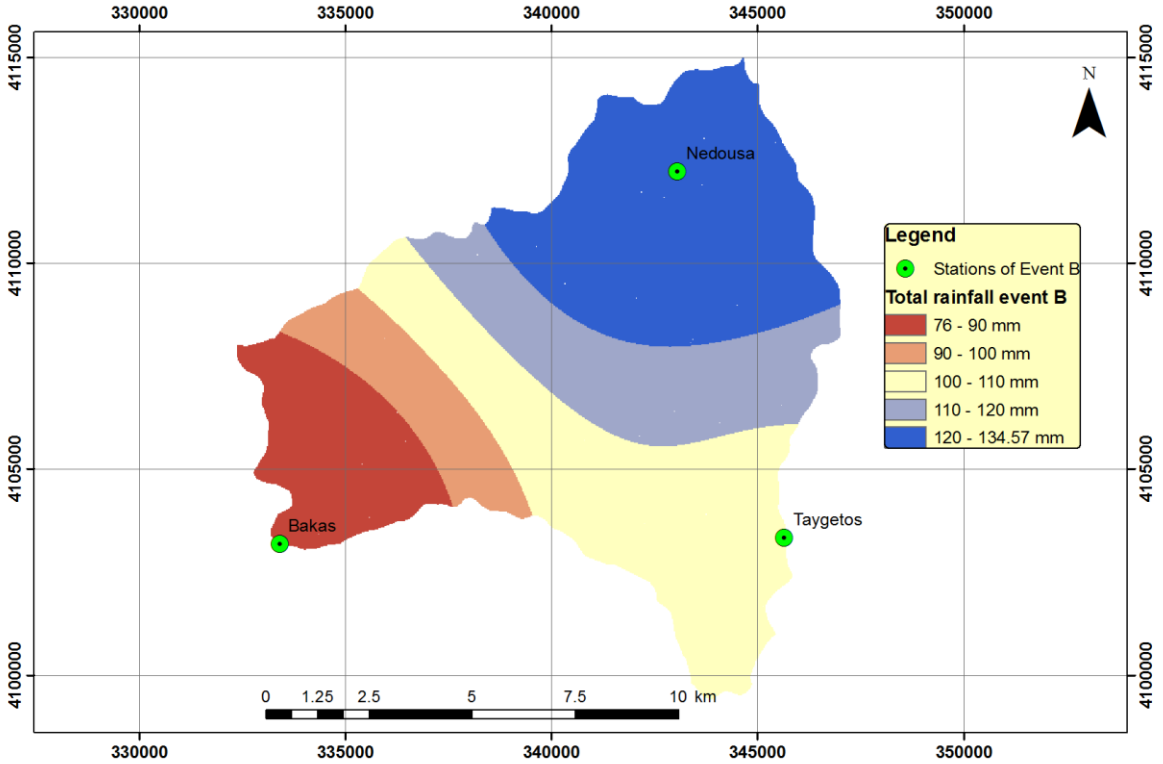


Figure 8.27: Total rainfall in mm of Event B.

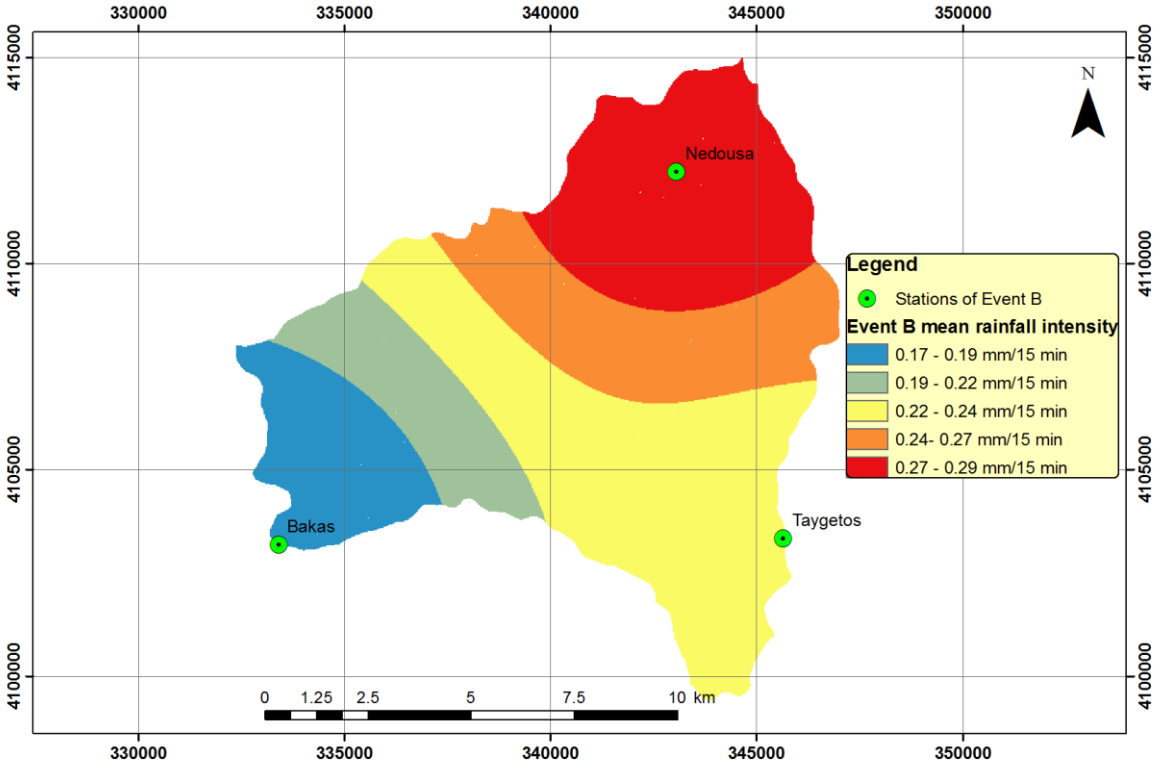


Figure 8.28: Mean intensity of rainfall, Event B.

The observed discharge for the 2st event measured in the Bakas Quarry gauge is illustrated in Figure 8.29. Runoff values up to 10/02/12 are presented.

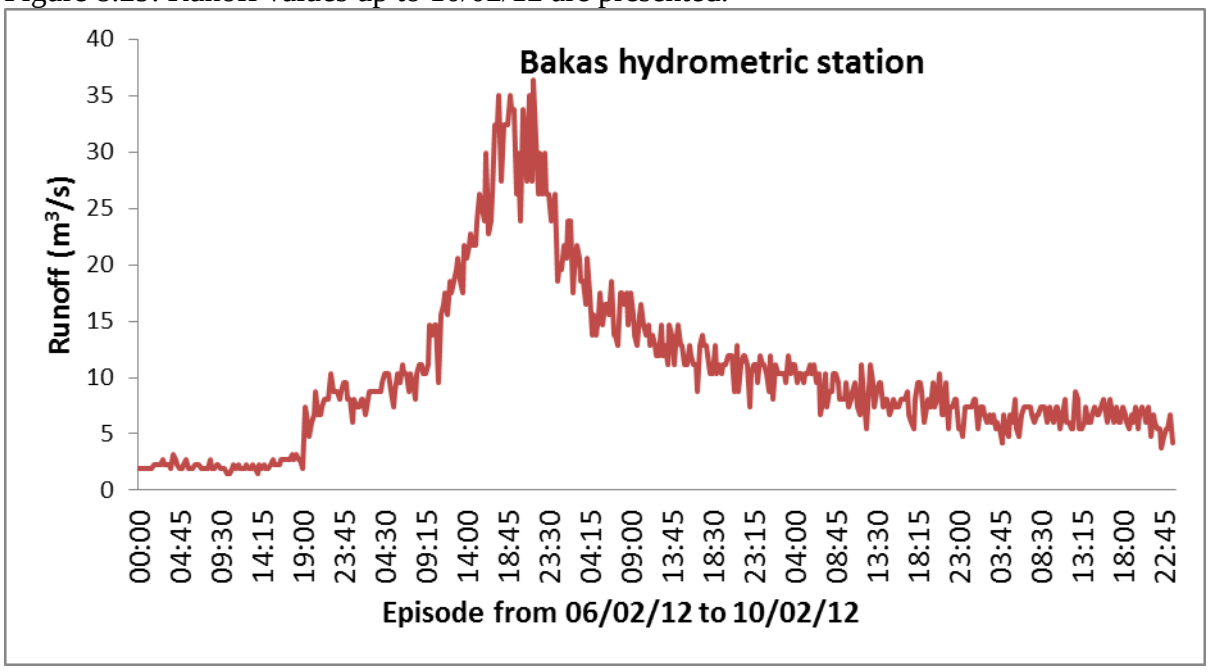


Figure 8.29: Observed discharge at Bakas Quarry gauge for Event B.

9 Simulation of flood events

In this chapter the results of the analyses conducted in the context of this study will be presented. The analyses involve the implementation of the proposed models and its comparison to the results obtained by the observed data. In order to calculate the performance of the two models four performance metrics are computed. The calculated hydrographs as well as the performances metrics are presented. For convenience, hereafter the rainfall event of 16/01/13 will be referred to as event A, while the event of 06/02/12 as event B.

9.1 Implementation of a lumped model to extract interflow discharge

In this section, results from a conceptual lumped model (refer to section 5.5.2) are presented in order to benchmark the distributed surface and complete models, while also providing an estimation of the interflow discharge. This is essential for surface runoff separation from the total hydrograph, before calibrating the surface model.

In this lumped configuration, the average CN value of the basin is also a calibration parameter, and the relative lag hysterisis of interflow and surface flow ($\delta - \tau$) is used. Rainfall in use in each time step is the average value over the whole basin.

9.1.1 Lumped model for event A

The parameter results from the lumped model configuration are presented in Table 9.1.

Table 9.1: Parameter results from lumped model (event A).

Parameter	Value
CN	41.3
λ	0.0745
κ	0.0004
μ	0
W_0	12.8 mm
$\delta - \tau$	2 hours
φ	0.0570

Table 9.2: Metric results from lumped model (event A).

Metric	Value
NSE	0.946
PEV	-22.1%
$PEPF$	+10.6%
ΔT_{PF}	+45 min

Given the generated timeseries surface runoff component of show in in Figure 9.1, the the total observed hydrograph

is calculated by removing the simulated interflow discharge. Surface runoff is shown in Figure 9.2.

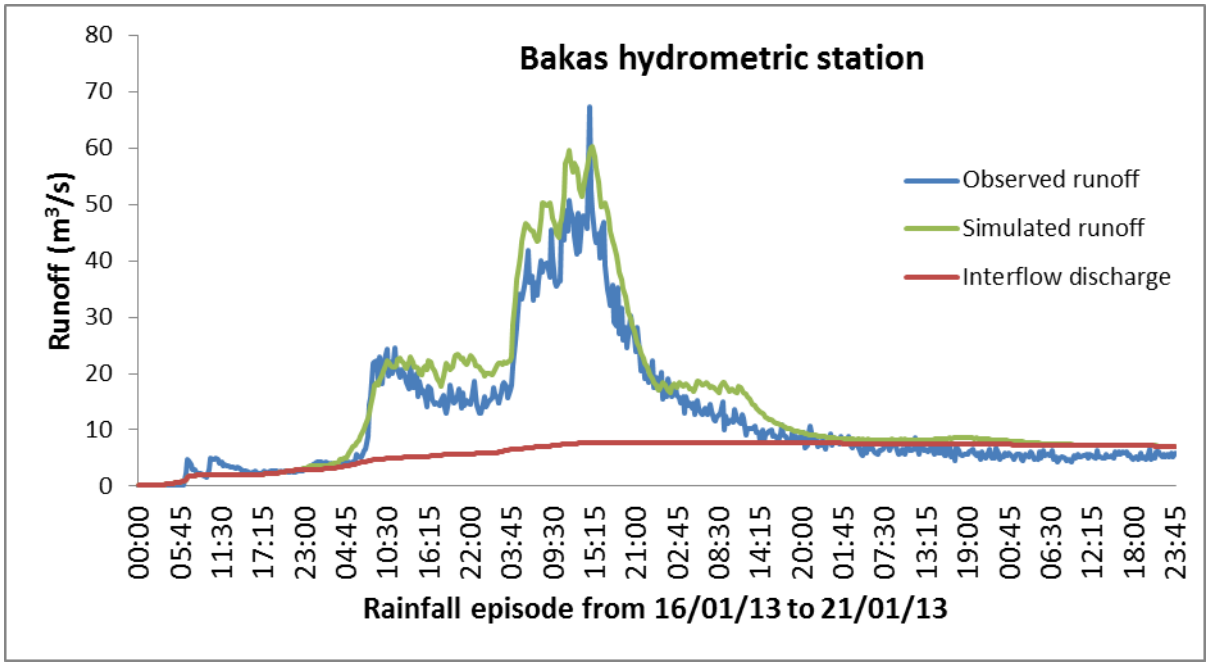


Figure 9.1: Lumped model results for event A.

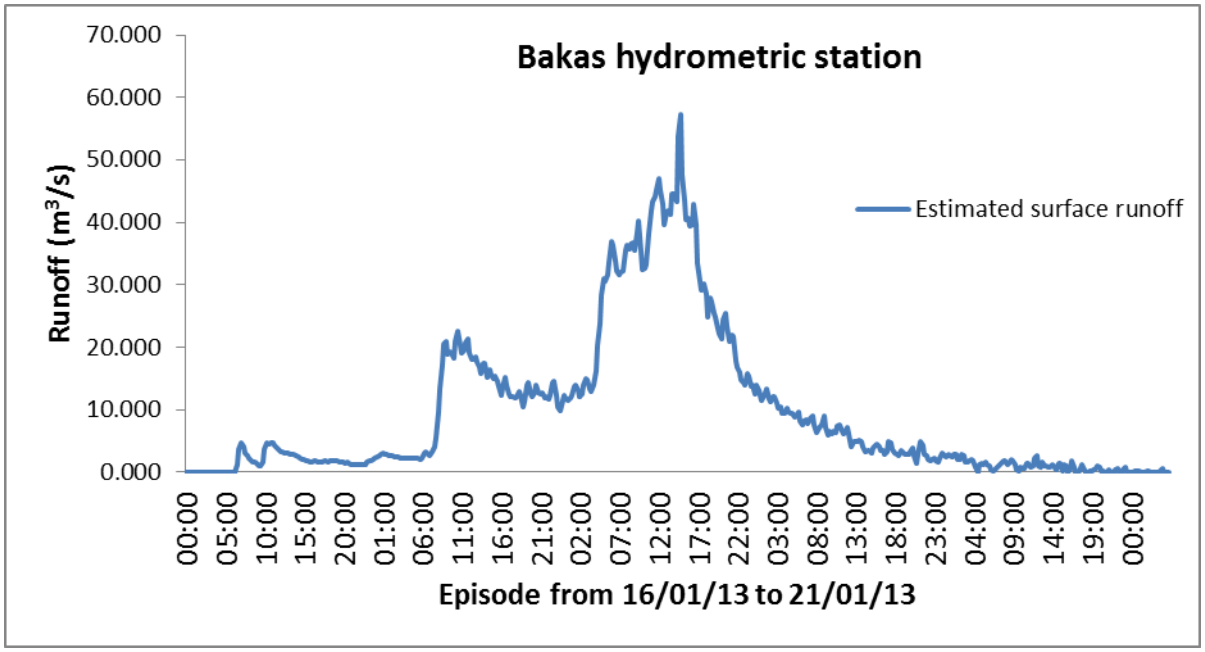


Figure 9.2: Estimated surface runoff of event A.

9.1.2 Lumped model for event B

The parameter results from the lumped model configuration are presented in Table 9.3.

Table 9.3: Parameter results from lumped model (event B).

Parameter	Value
CN	54.1
λ	0.0010
κ	0.0007
μ	0.0011
W_0	19.9 mm
$\delta - \tau$	2 hours
φ	0.0384

The performance of the lumped model is very satisfactory, as shown in Table 9.2. In particular, NSE coefficient is very high with a value of 0.946.

Table 9.4: Metric results from lumped model (event B).

Metric	Value
NSE	0.957
PEV	-0.34%
$PEPF$	+7.76%
ΔT_{PF}	+105 min

Given the generated timeseries showin in Figure 9.1, the surface runoff component of the total observed hydrograph is calculated, by removing the simulated interflow discarge. Surface runoff is shown in Figure 9.4.

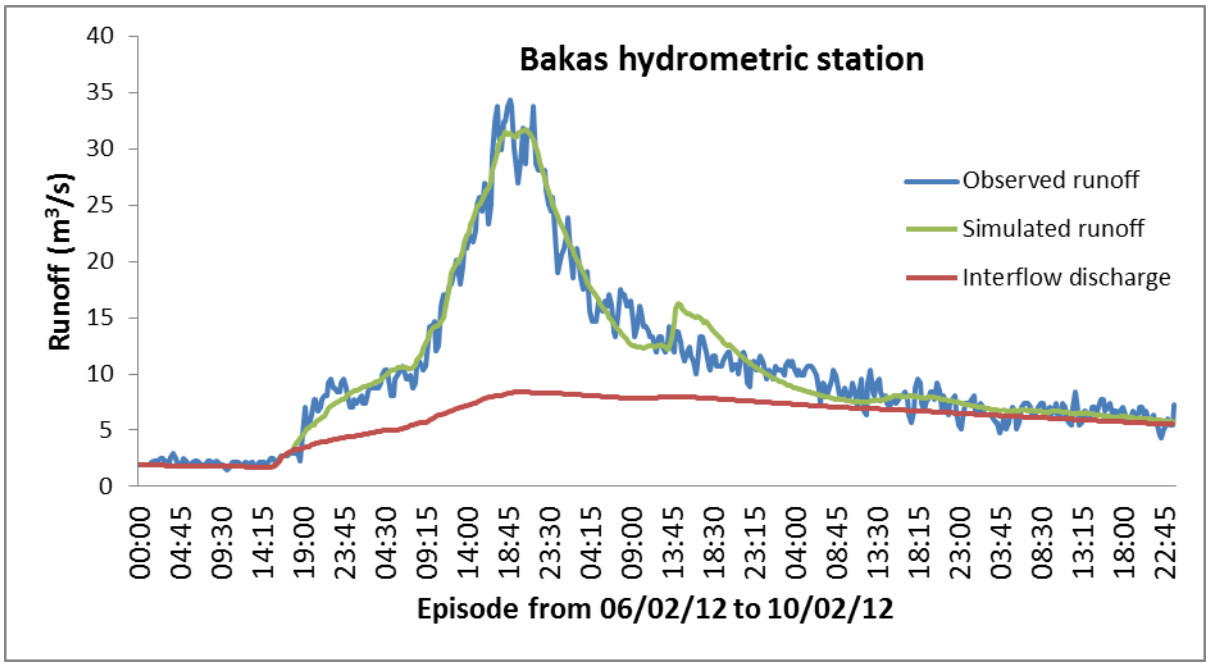


Figure 9.3: Lumped model results for event B.

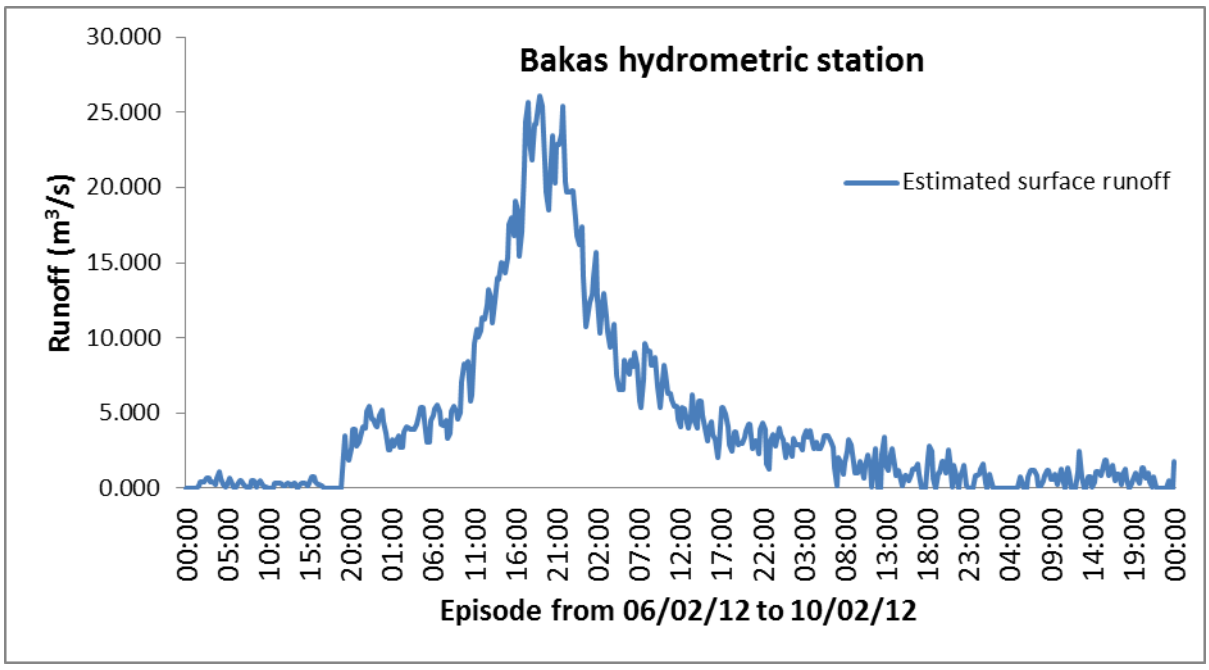


Figure 9.4: Estimated surface runoff of event B.

The timeseries of Figures 9.2 and 9.4 will be used as the observed surface flow of Events A and B respectively for the employment of the surface model.

9.2 Distributed surface model

9.2.1 Surface model, Event A

Optimized parameters for Event A are shown in Table 9.5. It is suggested by the AMC coefficient value that the antecedent moisture conditions are extremely dry. The initial abstraction ratio is also low at 5%, but in line with values in literature.

Table 9.5: Optimized parameters of surface model, Event A

Parameter	Value
λ	0.050
AMC	0.005

The performance metrics are shown in Table 9.6. The surface model exhibits good enough performance (given the uncertainty in interflow separation), with a NSE metric of 0.704. Peak flow estimation is satisfactory, as there the simulated peak is only 5.6% larger and comes 2 hours earlier. As seen by both the PEV metric and the timeseries form in Figure there is an overestimation of the volume. Note however that it is compared to estimated surface runoff volume, which may be misleading.

Table 9.6: Performance metrics of surface model, Event A

Metric	Value
NSE	0.050
PEV	-21.4%
$PEPF$	-5.57%
ΔT_{PF}	-120 min

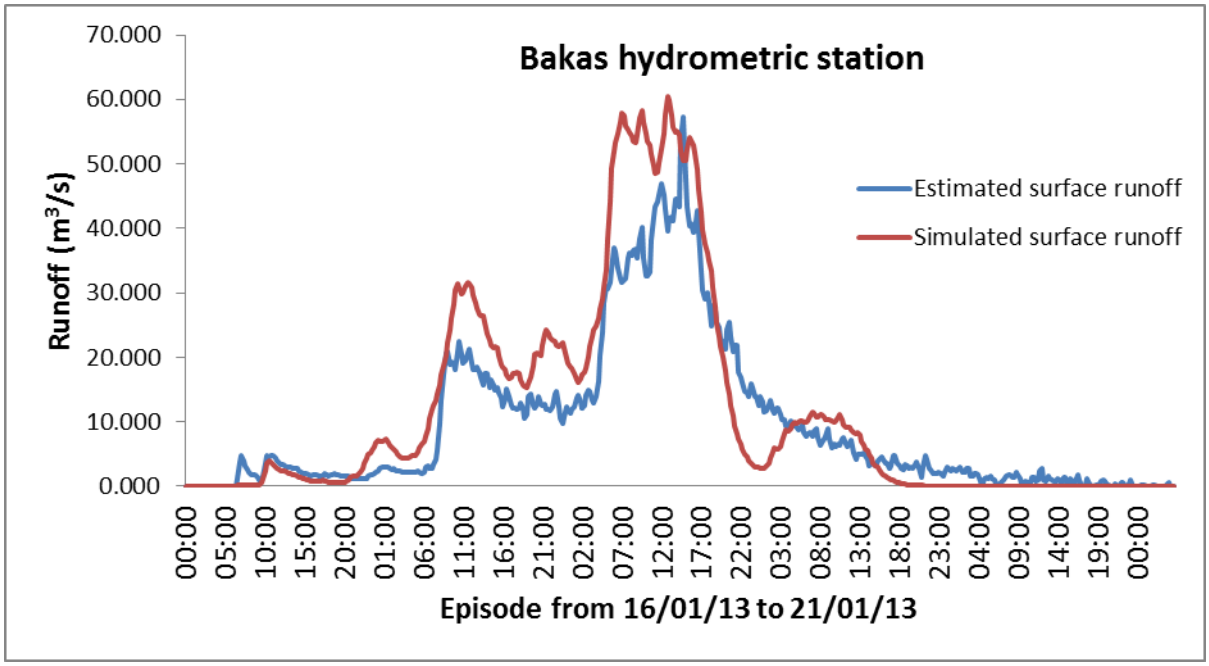


Figure 9.5: Simulated timeseries for Event A by the surface model.

Figure 9.6 illustrates the adjusted CN values. Nearly the whole area is below 60, with average value of 37.4. There also areas with CN values below 28, which means that these do not produce runoff. Figure 9.7 depicts overland velocity of the area and channel velocities, Overland velocity ranges between 0.003 m/s and 0.234 m/s. Channel velocity ranges between 1.09 m/s to 2.91 m/s. Concentration time t_c is equal to 3.69 hours.

Isochrones are shown in Figure 9.8. Most of the basin drains within 8 hours, however there are some areas in the east that need betweeen 8 and 16 hours. Mean travel time is 5.7 hours.

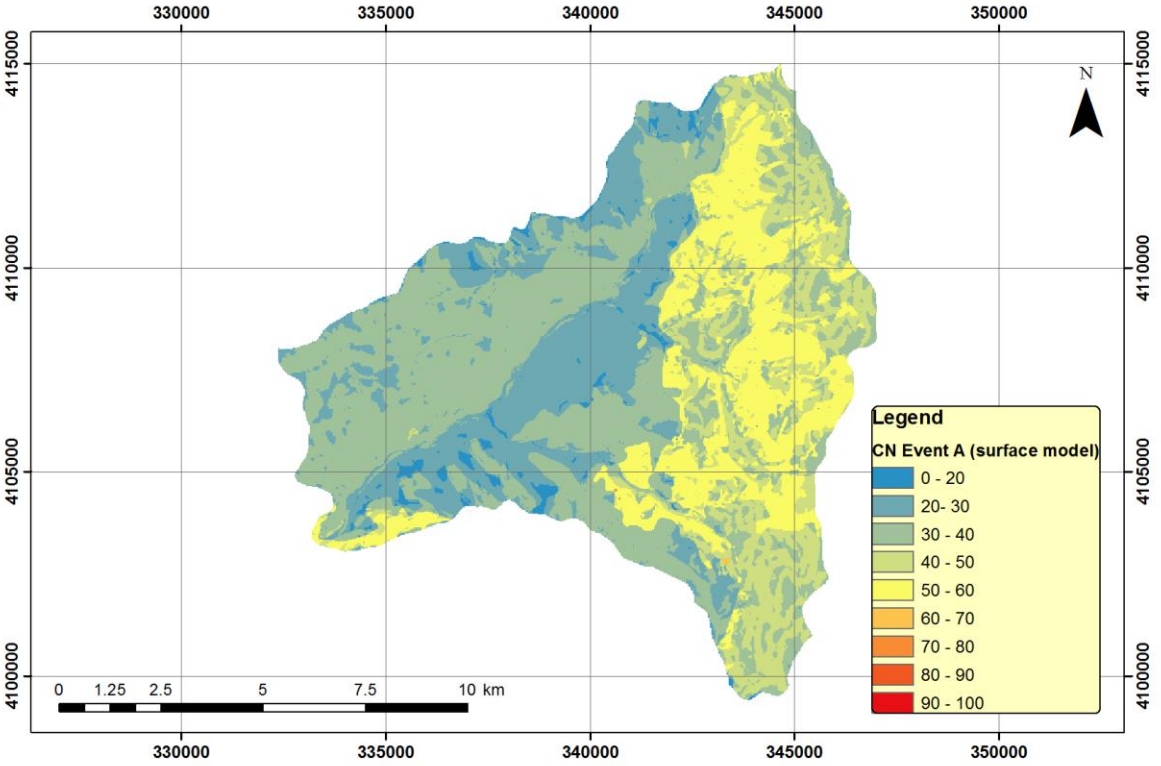


Figure 9.6: Adjusted CN values for event A (surface model).

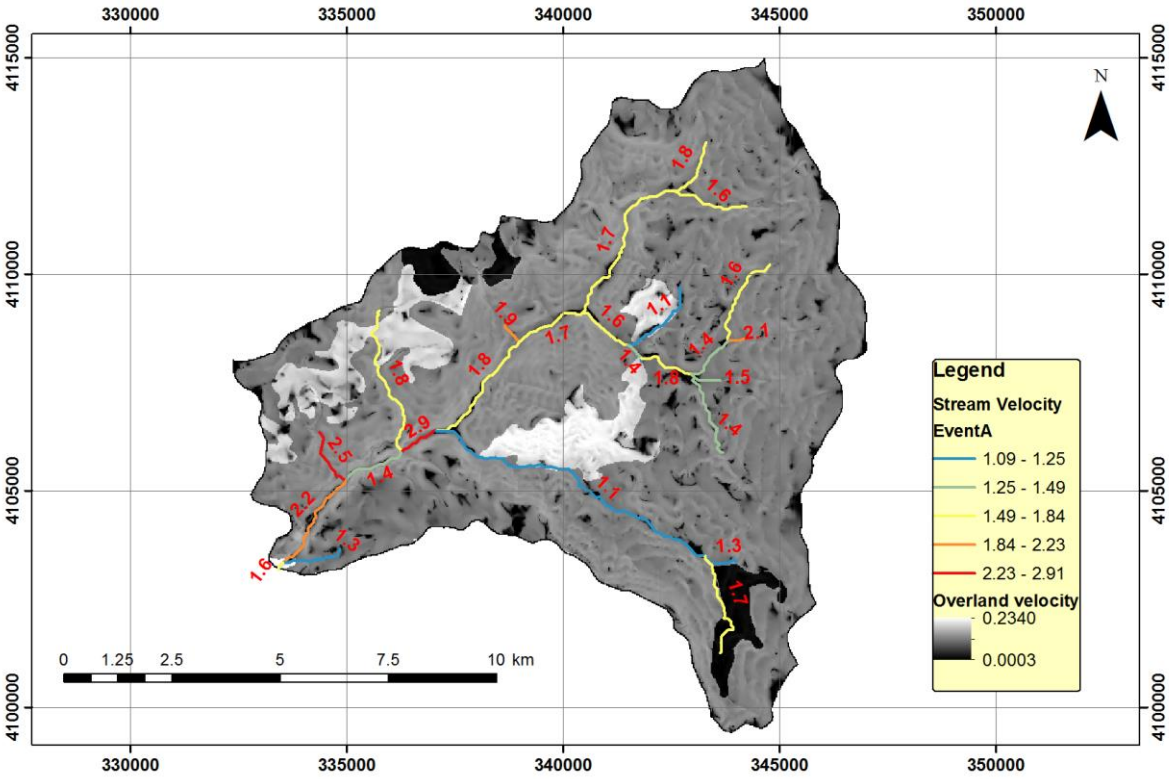


Figure 9.7: Overland and channel velocities of Event A.

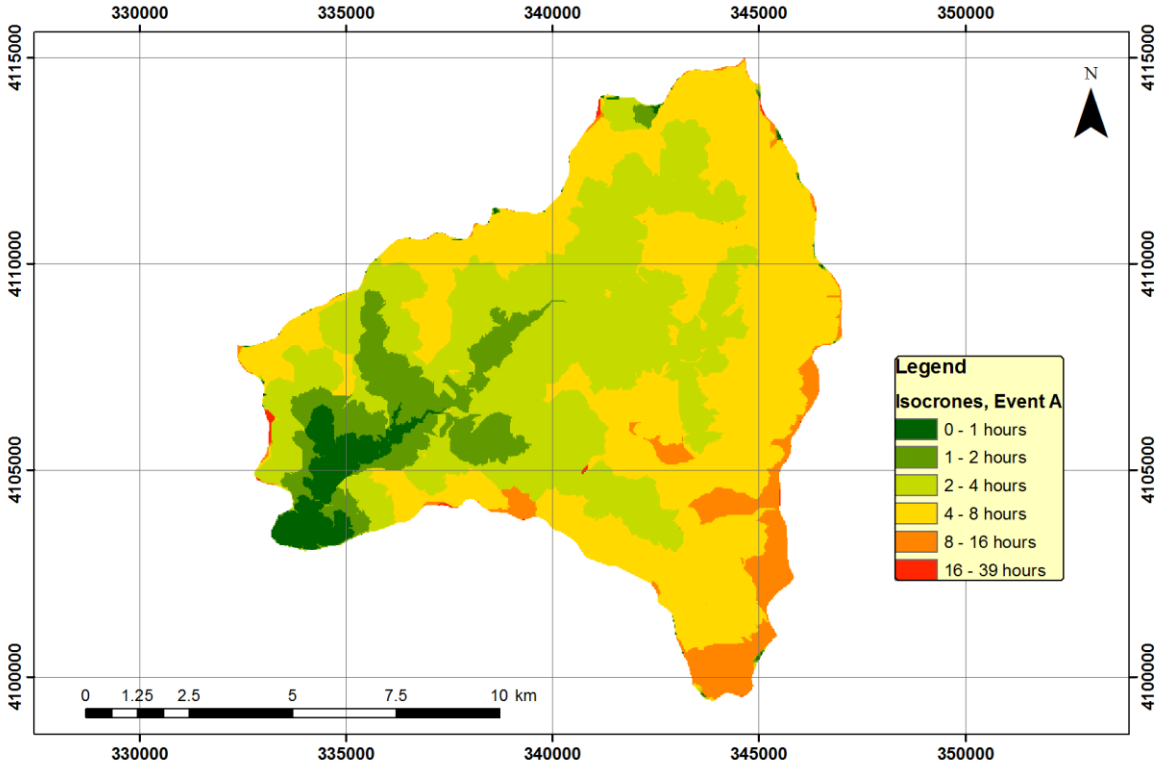


Figure 9.8: Isochrones of event A (surface model).

9.2.2 Surface model, Event B

Optimized parameters for Event B are shown in Table 9.7. The AMC coefficient is higher than the one of Event A, though the antecedent moisture conditions are dry in this case also. The initial abstraction ratio is again low at 1.1%.

Table 9.7: Optimized parameters of surface model, Event B

Parameter	Value
$\boldsymbol{\lambda}$	0.011
$\boldsymbol{AMC}$	0.233

The performance metrics are shown in Table 9.8. The surface model exhibits very good performance with a NSE metric of 0.901. Peak flow estimation is satisfactory, as the simulated peak is 9.62% lower (23.5 m^3/s vs 26.08 m^3/s) and comes 2 hours later. As seen by both the PEV metric and the timeseries form in Figure there is an underestimation of the volume. Note however that it is compared to estimated surface runoff volume, which may be misleading.

Table 9.8: Performance metrics of surface model, Event B

Metric	Value
$\boldsymbol{NSE}$	0.901
$\boldsymbol{PEV}$	9.62%
$\boldsymbol{PEPF}$	+14.99%
$\boldsymbol{\Delta T_{PF}}$	+120 min

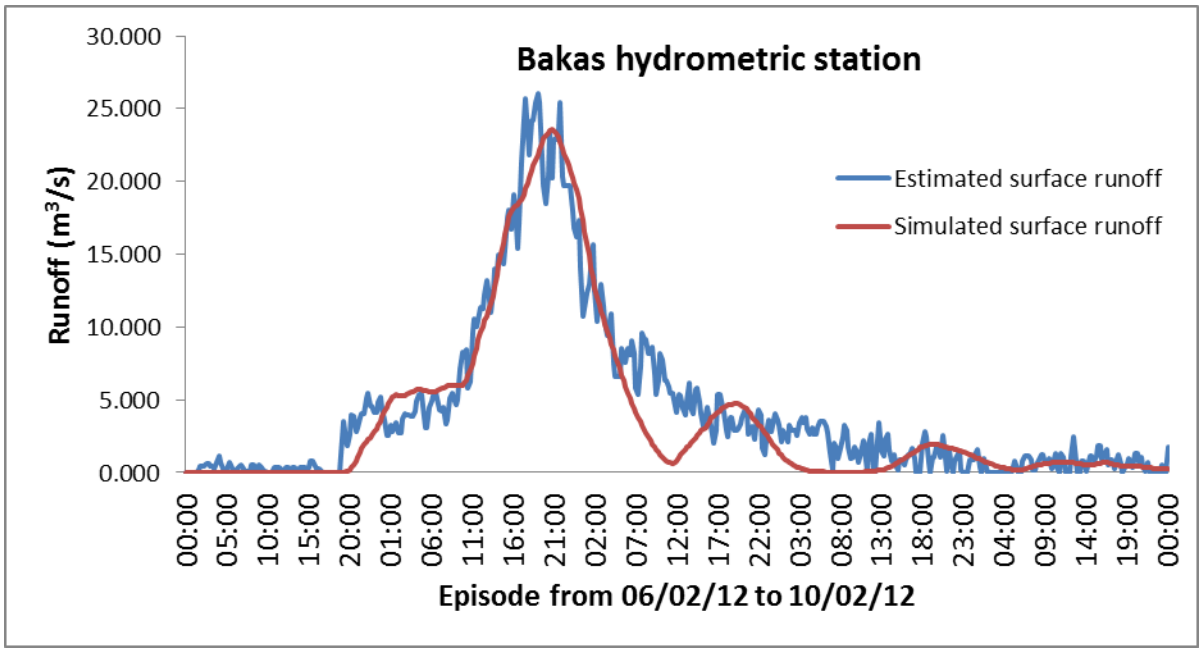


Figure 9.9: Simulated timeseries for Event B by the surface model.

Figure 9.10 illustrates the adjusted CN values by the AMC coefficient. The average CN value is 49, and there are many arears with values over 70, mainly in the east. Figure 9.11depicts overland velocity of the area and channel velocities. Channel velocity ranges between 0.9 m/s to 2.43 m/s, approximately at 94.3% of Event A channel velocities. Concentration time t_c is equal to 4.22 hours.Overland velocities remain the same, as the computation is not dependent on rainfall intensity (see equation (5.12)). The changed overall flow-time of the cells has a significant effect on the isochrones of Event B as seen in Figure 9.12. Mean travel time is 6.15 hours, a 7.8% increase.

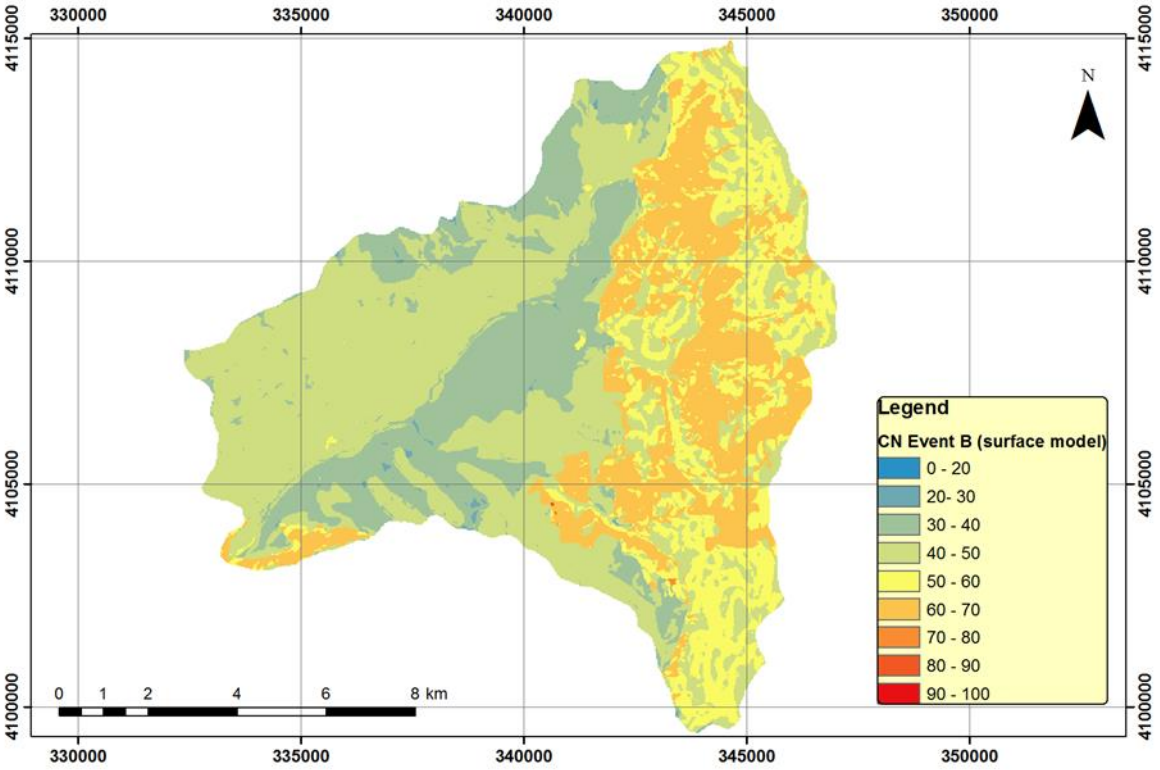


Figure 9.10: Adjusted CN values for event B (surface model).

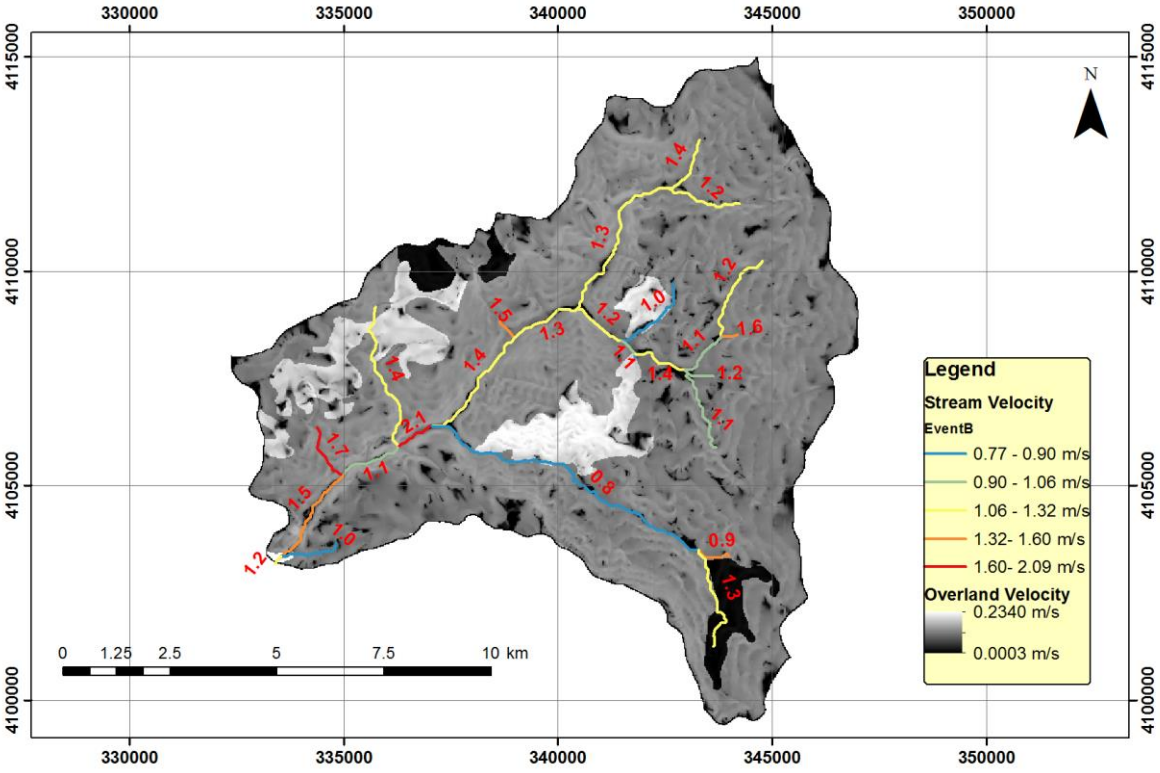


Figure 9.11: Overland and channel velocities of Event B (surface model).

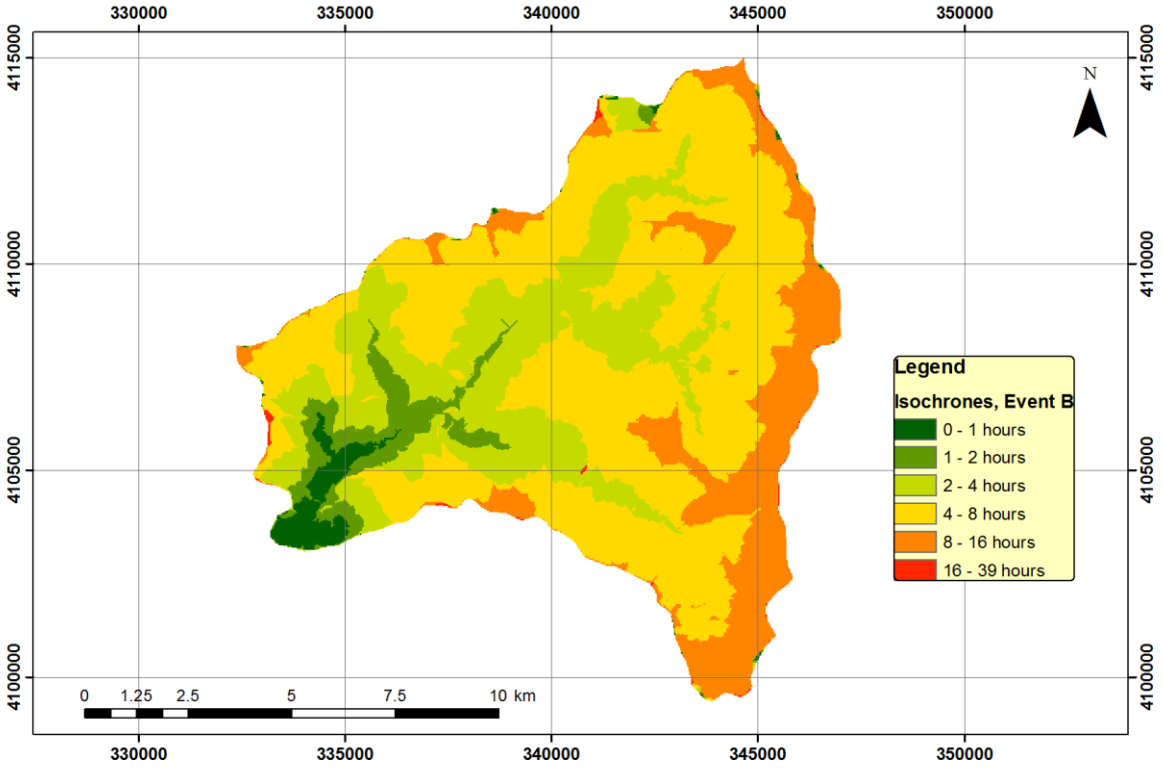


Figure 9.12: Isochrones of event B (surface model).

9.3 Distributed complete model

9.3.1 Complete model, Event A

Optimized parameters for Event A are shown in Table 9.9. The AMC coefficient is higher than the one of the surface model, and in contrast the initial abstraction ratio is much higher at 0.245. By the comparison between μ and κ parameters, it is suggested that interflow is an order of magnitude lower than percolation. Initial moisture W_0, lumped across the basin, is in line with the dry conditions suggested by the low AMC value.

Table 9.9: Optimized parameters of the complete model, event A.

Parameter	Value
λ	0.245
AMC	0.034
κ	0.00078
μ	0.0061
W_0	16.50 mm
δ	9.25 hours

Performance of the complete model is satisfactory, as implied by the NSE value of 0.865 (Table 9.10). The difference between peak flows is marginal, as the simulated hydrograph has a 3% lower peak (65.26 m^3/s versus 67.34 m^3/s) which comes 1 hour earlier. The difference in volumes produced is +15.4% (higher in the simulated hydrograph). As Figure 9.13 suggest, interflow discharge component in the simulated hydrograph resembles the observed one.

Table 9.10: Performance metrics of complete model, Event A.

Metric	Value
NSE	0.865
PEV	-15.4%
$PEPF$	+3.0%
ΔT_{PF}	-1 hour

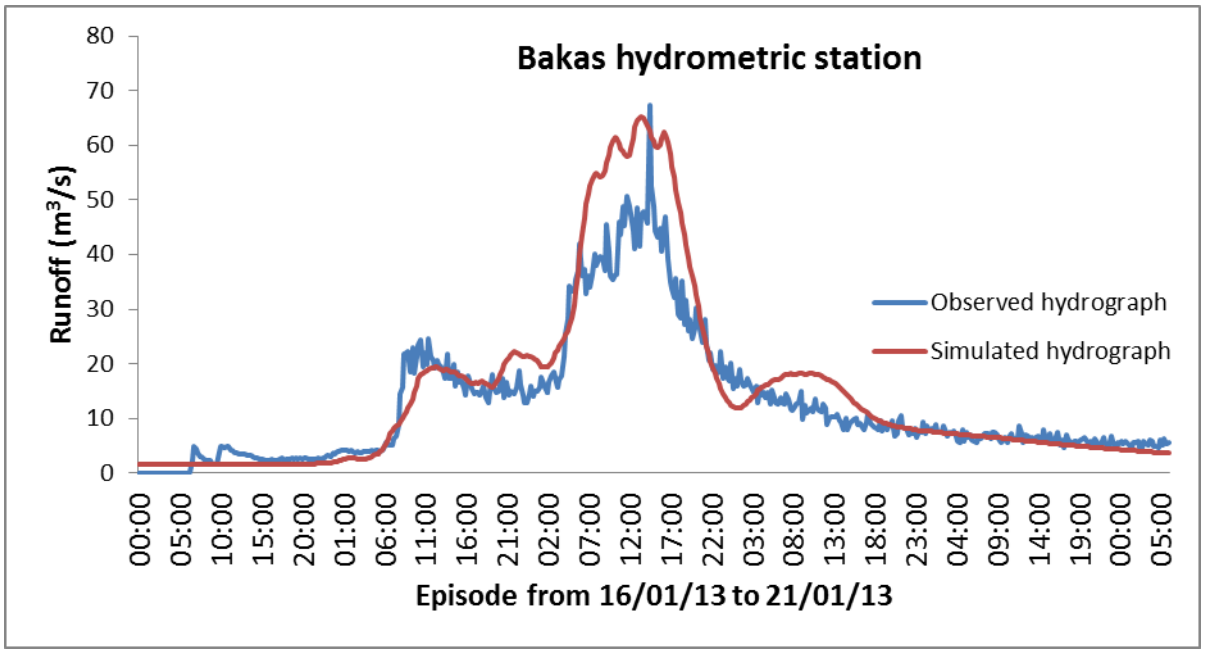


Figure 9.13: Hydrograph generated by the complete model, Event A.

As runoff intensity is higher considering the total hydrograph, there is an increase in channel velocities as seen in Figure 9.14, which now vary in the range of 1.02 – 3.39 m/s. Concentration time t_c is equal to 3.59 hours. This translates to slightly different isochrones, as seen in Figure 9.15. Mean travel time decreases to 5.34 hours.

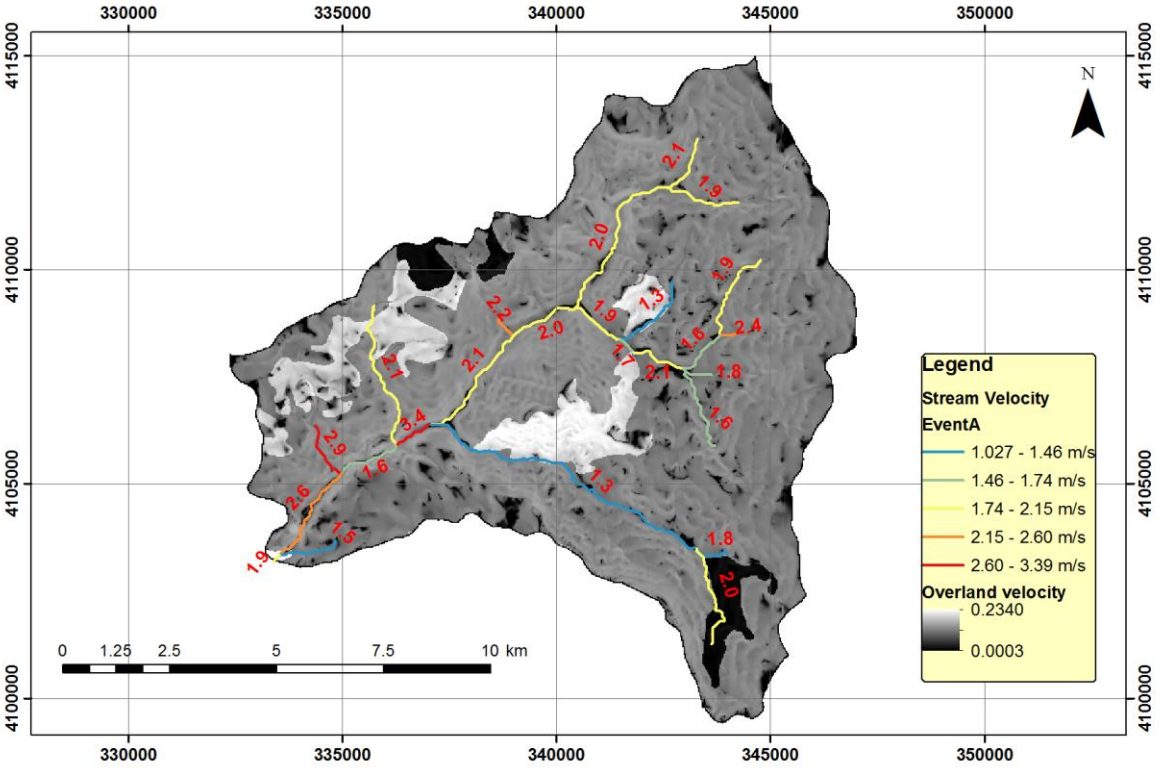


Figure 9.14: Channel and overland velocity, Event A, complete model.

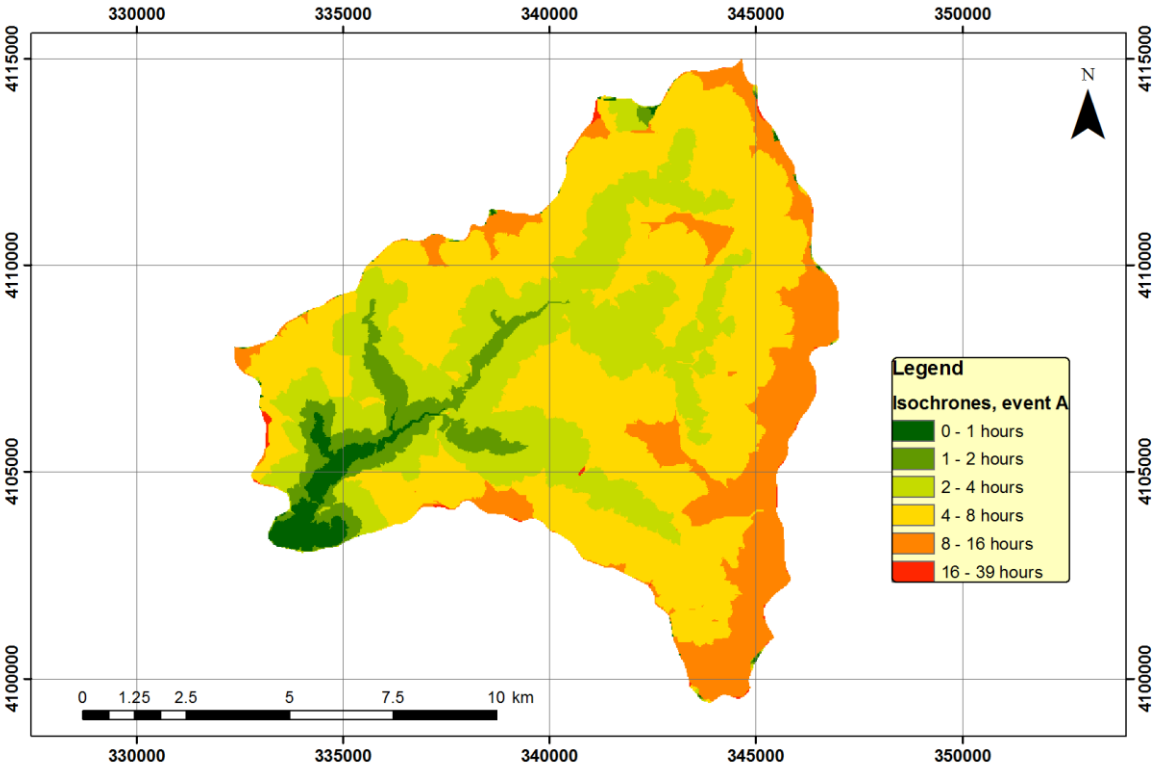


Figure 9.15: Isochrones for Event A, complete model.

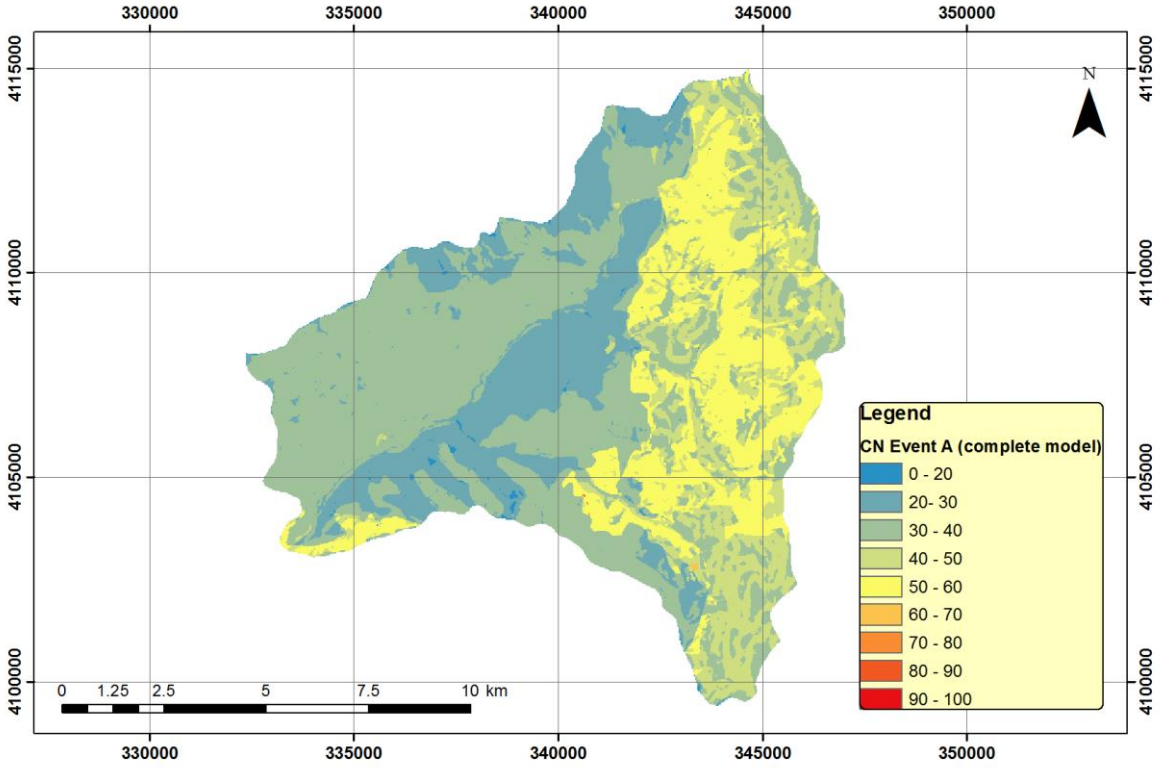


Figure 9.16: Adjusted CN values for event A (complete model).

9.3.2 Complete model, Event B

Optimized parameters for Event B are shown in Table 9.11. The AMC coefficient is lower than the one of the surface model for the same event. The initial abstraction ratio is lower (approximately zero). By the comparison between μ and κ parameters, it is suggested that interflow is more significant in event B than in A. This is further indicated by the recession limbs of both the observed and simulated hydrographs in Figure. Initial moisture W_0 is roughly the same as in Event A, which may be explained by the fact that antecedent moisture conditions are dry for both events.

Table 9.11: Optimized parameters of the complete model, event B.

Parameter	Value
$\boldsymbol{\lambda}$	0.0005
$\boldsymbol{AMC}$	0.209
$\boldsymbol{\kappa}$	0.0012
$\boldsymbol{\mu}$	0.0038
$\boldsymbol{W_0}$	16.48 mm
$\boldsymbol{\delta}$	15.5 hours

Performance of the complete model is exceptional as implied by all metrics of Table 9.12. The NSE value of 0.946 can be considered excellent. The difference between peak flows is small, as the simulated hydrograph has a 7.6% lower peak (31.36 m^3/s versus 33.96 m^3/s) which comes 1 hour later. The difference in volumes produced is marginal at +0.38% (higher in the observed hydrograph). As suggest, interflow discharge component in the simulated hydrograph resembles the observed one.

Table 9.12: Performance metrics of complete model, Event B.

Metric	Value
NSE	0.865
PEV	-15.4%
$PEPF$	+3.0%
ΔT_{PF}	+1 hour

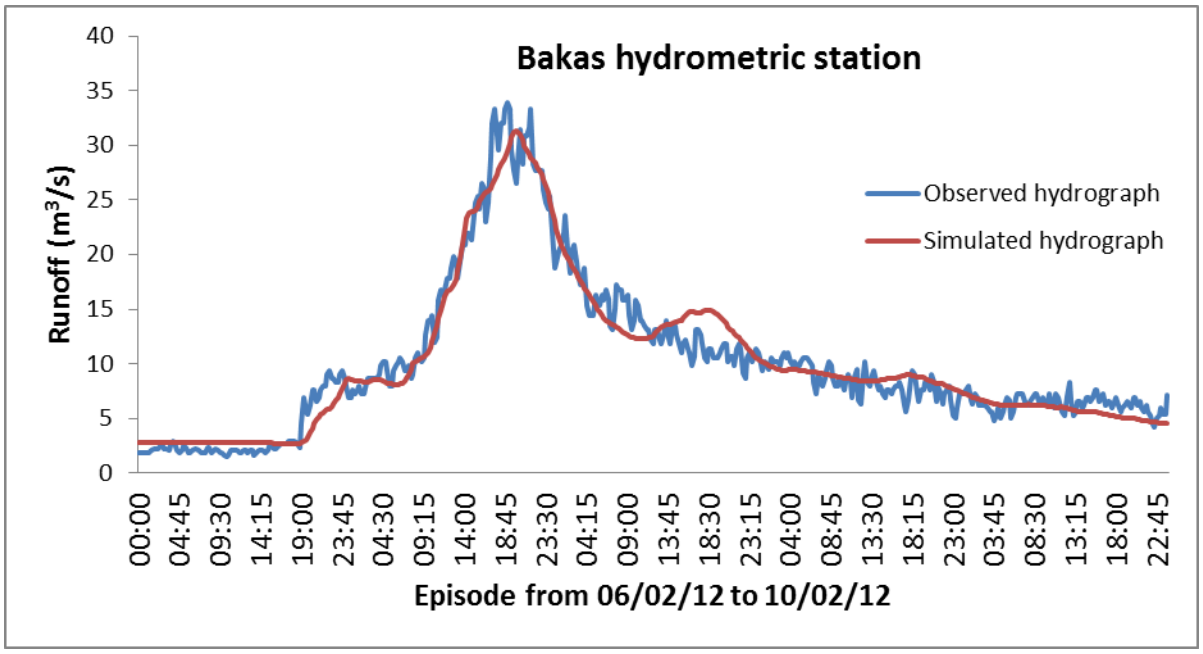


Figure 9.17: Hydrograph generated by the complete model, Event B.

As runoff intensity is higher considering the total hydrograph, there is an increase in channel velocities as seen in Figure 9.18, which range between 0.91 – 2.43 m/s. Concentration time t_c is equal to 3.77 hours.This translates to different isochrones from the surface model, as seen in Figure 9.19. Mean travel time decreases to 5.84 hours.

The adjusted CN values by the AMC coefficient can be visualized in Figure 9.20. There is consistency in the both in the complete and the surface model, as Event B's CN values are much higher than A's.

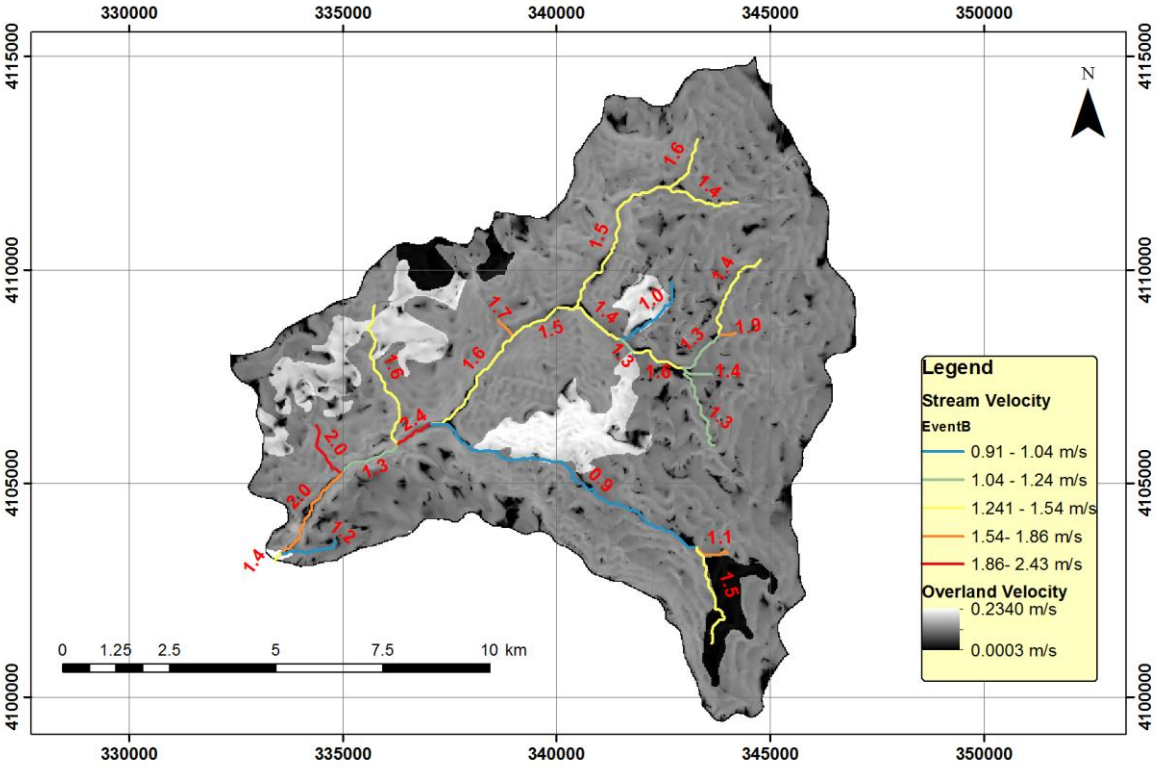


Figure 9.18: Channel and overland velocities, Event B, complete model.

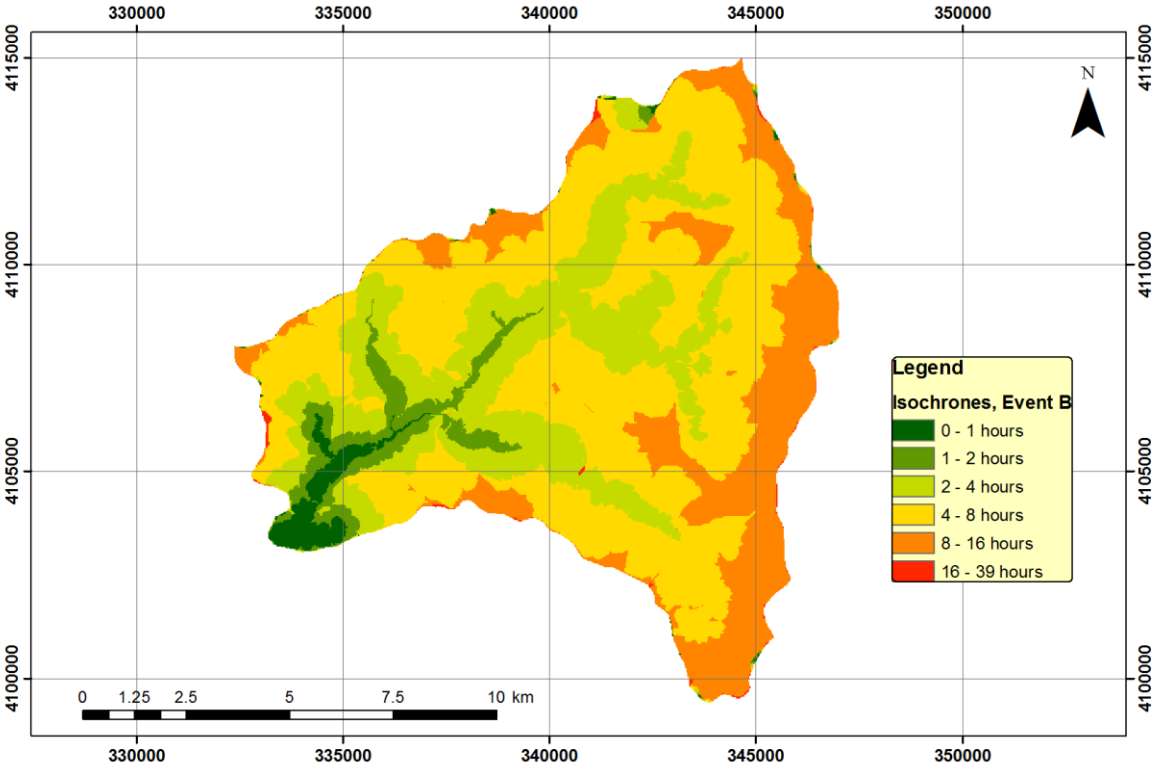


Figure 9.19: Isochrones for Event B, complete model.

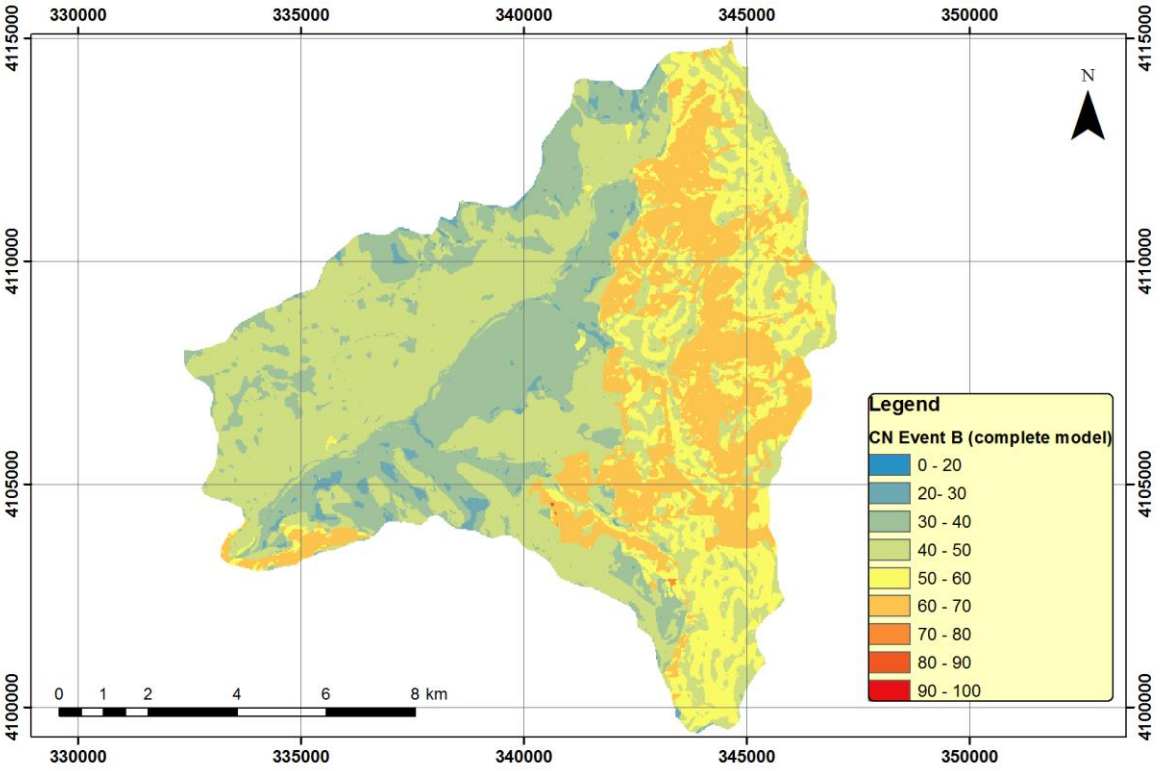


Figure 9.20: Adjusted CN values for event B (complete model).

9.4 Model assessment

As evident by results, the complete model has the best performance, even though it needs four more parameters in order to operate. It is also comparable in efficiency to the empeirical 8-parameter semi-black box lumped model used to separate interflow, which is quite an achievement as the complete model tries to spatially aproximate natural procedures. However, the surface model gives good enough results, especially considering that Nedontas is a karstified basin, thus interflow being a significant component of the hydrograph. Separation of interflow is a difficult task, ridden with uncertainty, but in cases where interflow is not pronounced or good quality surface flow data are provided the surface model may suffice.

Both models exhibit worse performance in Event A, which has according to optimized results from both models extremely dry antecedent conditions. Parameter AMC seems to be consistent between surface and complete model in both events, and is slightly lower within the complete model. Paramer λ, the initial abstraction ratio, however is not consistent within the different implementations. There is the possibility of underoptimization in event A, as its impact in the complete model (where it exhibits a very high value of 24.5%) is minimal compared to AMC, k and μ parameters, where the model is most sensitive.

Channel velocity estimation is consistent across events and model implementation, as Event A exhibits faster values while being of much higher rainfall/runoff intensity.

10 Conclusions

10.1 Research and technological outcomes

The scope of the thesis spreads across two domains:

In the **research domain**, the aim is to present recent advances in flood hydrology within a distributed event-based modelling framework, including:

- a GIS-based approach for extracting distributed maps of the so-called reference CN (corresponding to average antecedent soil moisture conditions and 20% initial abstraction ratio), accounting for three key physiographic characteristics of the river basin, i.e. permeability, vegetation/land cover and drainage capacity (slope);
- an empirical formula for adjusting CN to any antecedent soil moisture conditions, expressed in terms a dimensionless parameter;
- a procedure for adjusting CN against any initial abstraction ratio;
- an approximate implementation of the concept of the varying time of concentration within runoff routing.

The key novelty of this work focused on routing procedures, for which we have developed an improved time-area approach. The original issue involves the assignment a spatially-varying channel velocities, which are estimated from easily-retrieved information, i.e. river network geometry (length, slope) and macroscopic estimations of roughens coefficients. Moreover, by considering the time of concentration of the basin within the hydraulic radius term, we can also vary the velocities according to the average intensity of each flood event, thus also accounting for flow conditions without employing a hydraulic model.

Significant advantage of the proposed model is its parsimonious formulation. In particular, the model only uses a lumped parameter for initial conditions and another lumped parameter for initial hydrological losses. Actually, the initial conditions can be quite safely estimated from the cumulative rainfall few days before the flood event, while a representative value for initial losses ratio can be also estimated on the basis of past flood events. Under this premise, our scheme can also run without needing calibration, since all rest model inputs can be extracted from morphological and physiographic data.

Another novelty involved the coupling the distributed model, for the generation and propagation of surface runoff, with a lumped component to represent the subsurface flow through a soil moisture accounting tank. This scheme ensures a more realistic description of the flood hydrograph, since it explicitly accounts for its two major components, i.e. overland flow and interflow. However, it makes essential the uses of additional parameters, which have to be inferred through calibration.

In the **technological domain** this work aims to bring together GIS tools and Python scientific packages, towards creating modern computational tools with augmented capabilities in data handling, data pre-processing, geo-spatial analysis, hydrological simulation, optimization and visualization of results.

Given the satisfactory results from the rainfall-runoff models that are employed, it is suggested that the proposed framework achieves its research aim. Particularly, the velocity-based method produces ensure quite realistic channel velocities, without being overly complex in implementation or data demanding, as the coarse estimation of roughness in the stream segments and the concertation time suffice for efficient hydrograph routing. The distributed

models employed, showcase relatively fast (in execution time) and very good (in the hydrological essence) performance.

The GUI, which has been developed as part of an integrated software for data handling, model simulation/optimization and visualization of model results, fulfills the second aim, and paves the way for more advanced applications.

10.2 Proposals for future research

Arguably, this work is just a primer for more advanced research. In particular:

- Multiple flood events across multiple basins with diferent characteristics should be tested within this modelling framework, to further explore the robustness of the tool, in terms of predictive capacity and computational performance.
- A key suggestion is to explore the possibility of incorporating a dynamic adjustment of the time of concertation within the simulated flood event (i.e. the basin time concertation time varies not only between event, but varies within the single event under study), thus allowing channel velocities to follow the variability of runoff.
- A challenging issue is the possibility of coupling a distributed rainfall-runoff model, to represent the lateral inflows to the stream network, with a hydraulic model, to analytically represent the routing processes. A comparison between the conceptual, velocity-based approach, with the analytic hydraulic model will yield interesting results concerning the battle of "performance versus parsimony in parameters and data".
- It is also worth exploring whether segmenting a basin to smaller entities given more observed hydrographs from hydrometric stations across the network will enhance performance, particularly regarding the segmenting of the lumped generalization of the underground linear reservoir for interflow generation to more discrete elements.

Finally, the user-perspective can be greatly enhanced with further developing the GUI of the software and adding more capabilities and automatizations.

References

Abbott, M. B., Bathurst, J. C., et al. (1986) 'An introduction to the european hydrological system - systeme hydrologique europeen, "SHE", 2: Structure of a physically - based, distributed modelling system', Journal of Hydrology, 87, pp. 61–77.

Abbott, M. B., Bathurst, J. C., et al. (1986a) 'An introduction to the European Hydrological System — Systeme Hydrologique Europeen, "SHE", 1: History and philosophy of a physically-based, distributed modelling system', Journal of Hydrology, 87(1–2), pp. 45–59. doi: 10.1016/0022-1694(86)90114-9.

Abbott, M. B., Bathurst, J. C., et al. (1986b) 'An introduction to the European Hydrological System — Systeme Hydrologique Europeen, "SHE", 2: Structure of a physically-based, distributed modelling system', Journal of Hydrology, 87(1–2), pp. 61–77. doi: 10.1016/0022-1694(86)90115-0.

Anaconda, I. (2018) 'Anaconda Distribution'. Available at: https://www.anaconda.com/distribution/.

Antoniadi, S. (2016) Investigation of the time - response variability of river basins. National Technical University of Athens.

Baltas, E., Dervos, N. and Mimikou, M. (2007) 'Research on the initial abstraction – storage ratio and its effect on hydrograph simulation at a watershed in Greece', Hydrology and Earth System Sciences Discussions, European Geosciences Union, 2007, 4 (4), pp. 2169-2204.

Bárdossy, A., Bronstert, A. and Merz, B. (1995) '1-, 2- and 3-dimensional modeling of water movement in the unsaturated soil matrix using a fuzzy approach', Advances in Water Resources, 18(4), pp. 237–251. doi: 10.1016/0309-1708(95)00009-8.

Bathurst, J. . et al. (2004) 'Validation of catchment models for predicting land-use and climate change impacts. 3. Blind validation for internal and outlet responses', Journal of Hydrology, 287(1–4), pp. 74–94. doi: 10.1016/j.jhydrol.2003.09.021.

Bathurst, J. C. (1986a) 'Physically-based distributed modelling of an upland catchment using the Systeme Hydrologique Europeen', Journal of Hydrology, 87(1–2), pp. 79–102. doi: 10.1016/0022-1694(86)90116-2.

Bathurst, J. C. (1986b) 'Sensitivity analysis of the Systeme Hydrologique Europeen for an upland catchment', Journal of Hydrology, 87(1–2), pp. 103–123. doi: 10.1016/0022-1694(86)90117-4.

Bathurst, J. C. and Cooley, K. R. (1996) 'Use of the SHE hydrological modelling system to investigate basin response to snowmelt at Reynolds Creek, Idaho', in Journal of Hydrology, pp. 181–211. doi: 10.1016/S0022-1694(96)80011-4.

Betson, R. P. (1964) 'What is watershed runoff?', Journal of Geophysical Research. doi: 10.1029/JZ069i008p01541.

Beven, J. K. (1985) ‘Distributed Models’, in Hydrological Forecasting. Wiley, pp. 405–435.

Beven, J. K. (2012) Rainfall–Runoff Modelling, Wiley - Blackwell. doi: 10.1002/ldr.630.

Beven, K. (1981) ‘Kinematic subsurface stormflow’, Water Resources Research, 17(5), pp. 1419–1424. doi: 10.1029/WR017i005p01419.

Beven, K. (1991) ‘Spatially Distributed Modeling: Conceptual Approach to Runoff Prediction’, in Recent Advances in the Modeling of Hydrologic Systems. Dordrecht: Springer Netherlands, pp. 373–387. doi: 10.1007/978-94-011-3480-4_17.

Beven, K. (1995) ‘Linking parameters across scales: Subgrid parameterizations and scale dependent hydrological models’, Hydrological Processes, 9(5–6), pp. 507–525. doi: 10.1002/hyp.3360090504.

Beven, K. (1997) Distributed Hydrological Modelling: Applications of the Topmodel Concept. John Wiley & Sons.

Binley, A., Beven, K. and Elgy, J. (1989) ‘A physically based model of heterogeneous hillslopes: 2. Effective hydraulic conductivities’, Water Resources Research, 25(6), pp. 1227–1233. doi: 10.1029/WR025i006p01227.

Bonell, M. et al. (1998) ‘High Rainfall, Response-Dominated Catchments: A Comparative Study of Experiments in Tropical Northeast Queensland with Temperate New Zealand’, in Isotope Tracers in Catchment Hydrology. Elsevier, pp. 347–390. doi: 10.1016/B978-0-444-81546-0.50018-5.

Bronstert, A. and Plate, E. J. (1997) ‘Modelling of runoff generation and soil moisture dynamics for hillslopes and micro-catchments’, Journal of Hydrology, 198(1–4), pp. 177–195. doi: 10.1016/S0022-1694(96)03306-9.

Chow, Maidment, D. R. and Mays, L. W. (1988) Applied Hydrology, Agriculture, Ecosystems & Environment. doi: 10.1016/j.soncn.2011.11.001.

Connolly, R. D. et al. (1997) ‘Distributed parameter hydrology model (Answers) applied to a range of catchment scales using rainfall simulator data. IV Evaluating pasture catchment hydrology’, Journal of Hydrology, 201(1–4), pp. 311–328. doi: 10.1016/S0022-1694(97)00052-8.

Connolly, R. D. and Silburn, D. M. (1995) ‘Distributed parameter hydrology model (ANSWERS) applied to a range of catchment scales using rainfall simulator data II: Application to spatially uniform catchments’, Journal of Hydrology, 172(1–4), pp. 105–125. doi: 10.1016/0022-1694(95)02741-7.

D. B. Beasley, L. F. Huggins and E. J. Monke (1980) ‘ANSWERS: A Model for Watershed Planning’, Transactions of the ASAE, 23(4), pp. 0938–0944. doi: 10.13031/2013.34692.

Deshmukh, D. S. et al. (2013) 'Estimation and comparision of curve numbers based on dynamic land use land cover change, observed rainfall-runoff data and land slope', Journal of Hydrology, 492, pp. 89–101. doi: 10.1016/j.jhydrol.2013.04.001.

Development, W. et al. (2000) Python Developer's Handbook.

Devia, G. K., Ganasri, B. P. and Dwarakish, G. S. (2015) 'A Review on Hydrological Models', in International Conference on Water Resources, Coastal and Ocean Engineering (ICWRCOE 2015), Aquatic Procedia, pp. 1001–1007. doi: 10.1016/j.aqpro.2015.02.126.

Diener, M. (2015) Python Geospatial Analysis Cookbook.

Dingman, S. L. (2002) Physical hydrology. 2nd edn. Prentice Hall.

Doe, W., Saghafian, B. and Julien, P. Y. J. (1996) 'Land-Use Impact on Watershed Response: the Integration of Two-Dimensional Hydrological Modelling and Geographical Information Systems', Hydrological Processes, 10(11), pp. 1503–1511.

Downer, C. W. et al. (2002) 'Theory, development, and applicability of the surface water hydrologic modelCASC2D', Hydrological Processes, 16(2), pp. 255–275. doi: 10.1002/hyp.338.

Downer, C. W. et al. (2006) 'Gridded Surface/Subsurface Hydrologic Analysis (GSSHA) Model: A Model for Simulating Diverse Streamflow-Producing Processes', in Watershed Models, pp. 131–157.

Downer, C. W. and Ogden, F. L. (2004) 'GSSHA: Model To Simulate Diverse Stream Flow Producing Processes', Journal of Hydrologic Engineering, 9(3), pp. 161–174. doi: 10.1061/(ASCE)1084-0699(2004)9:3(161).

Durand, P., Robson, A. and Neal, C. (1992) 'Modelling the hydrology of submediterranean montane catchments (Mont-Lozère, France) using TOPMODEL: initial results', Journal of Hydrology, 139(1–4), pp. 1–14. doi: 10.1016/0022-1694(92)90191-W.

Eckhardt, K. (2008) 'A comparison of baseflow indices, which were calculated with seven different baseflow separation methods', Journal of Hydrology, 352(1–2), pp. 168–173. doi: 10.1016/j.jhydrol.2008.01.005.

Eckhardt, K., Fohrer, N. and Frede, H.-G. (2005) 'Automatic model calibration', Hydrological Processes, 19(3), pp. 651–658. doi: 10.1002/hyp.5613.

Editors, S. et al. (2008) Differential Evolution - A Practical Approach to Global Optimization, New York. doi: 10.1162/evco.2004.12.2.269.

Efstratiadis, A., Koukouvinos, A., et al. (2014) Description of regional approaches for the estimation of characteristic hydrological quantities, DEUCALION – Assessment of flood flows in Greece under conditions of hydroclimatic variability: Development of physically-established conceptual-probabilistic. Athens.

Efstratiadis, A., Koussis, A. D., et al. (2014) 'Flood design recipes vs. reality: can predictions for ungauged basins be trusted?', Natural Hazards and Earth System Sciences, 14(6), pp. 1417–1428. doi: 10.5194/nhess-14-1417-2014.

Ewen, J., Parkin, G. and O'Connell, P. E. (2000) 'SHETRAN: Distributed River Basin Flow and Transport Modeling System', Journal of Hydrologic Engineering, 5(3), pp. 250–258. doi: 10.1061/(ASCE)1084-0699(2000)5:3(250).

Fangohr, H. (2004) 'A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering', in Computational, pp. 1210–1217. doi: 10.1007/978-3-540-25944-2_157.

Fangohr, H. (2015) 'Introduction to Python for Computational Science and Engineerin', eBook, p. 167.

Förster, K. et al. (2016) 'An open-source MEteoroLOgical observation time series DISaggregation Tool (MELODIST v0.1.1)', Geoscientific Model Development, 9(7), pp. 2315–2333. doi: 10.5194/gmd-9-2315-2016.

Haan, C., Barfield, B. and Hayes, J. (1994) Design Hydrology and Sedimentology for Small Catchments. New York: Academic Press.

Hall, F. R. (1968) 'Base-Flow Recessions-A Review', Water Resources Research, 4(5), pp. 973–983. doi: 10.1029/WR004i005p00973.

Hewlett, J. D. (1961) Watershed management, Report for 1961 Southeastern Forest Experiment Station. Ashville.

Hewlett, J. D. and Hibbert, A. R. (1967) 'Factors affecting the response of small watersheds to precipitation in humid areas', in International Symposium on Forest Hydrology. doi: 10.1177/0309133309338118.

Holko, L. et al. (2002) 'Groundwater runoff separation - test of applicability of a simple separation method under varying natural conditions', FRIEND 2002 - Regional Hydrology: Bridging the Gap between Research and Practice, 1(274), pp. 265–272. doi: 10.1109/ICoOM.2013.6626507.

Huang, M. et al. (2006) 'A modification to the Soil Conservation Service curve number method for steep slopes in the Loess Plateau of China', Hydrological Processes, 20(3), pp. 579–589. doi: 10.1002/hyp.5925.

Huggins, L. F. and Monke, E. J. (1968) 'A Mathematical Model for Simulating the Hydrologic Response of a Watershed', Water Resources Research, 4(3), pp. 529–539. doi: 10.1029/WR004i003p00529.

Izzard, C. F., Hicks and I., W. (1946) 'Hydraulics of runoff from developed surfaces', in 26th Annual Meetings of the Highway Research Board, pp. 129–150.

Kirchner, J. W. (2003) 'A double paradox in catchment hydrology and geochemistry', Hydrological Processes, 17(4), pp. 871–874. doi: 10.1002/hyp.5108.

Kirkby, M. J. (1975) Hydrograph Modelling Strategies. Department of Geography, University of Leeds.

Kirkby, M. J. (1997) 'TOPMODEL: A personal view', Hydrological Processes, 11(9), pp. 1087–1097. doi: 10.1002/(SICI)1099-1085(199707)11:9<1087::AID-HYP546>3.0.CO;2-P.

Kliner, K. and Knezek, M. (1974) 'The underground runoff separation method making use of the observation of ground water table', Hydrology and hydromechanics, 22(5), pp. 457–466.

Koutsoyiannis, D. and Xanthopoulos, T. (1999) Engineering Hydrology.

Lin, J. W. B. (2012) 'Why python is the next wave in earth sciences computing', Bulletin of the American Meteorological Society, 93(12), pp. 1823–1824. doi: 10.1175/BAMS-D-12-00148.1.

Linsley, R. et al. (1975) Hydrology for engineers. McGraw-Hill.

Maclaren, N. (2012) Why (and Why Not) to Use Fortran.

Mancusi, L., Albano, R. and Sole, A. (2015) 'FloodRisk : a QGIS plugin for flood consequences estimation', Geomatics Workbooks, pp. 483–496. doi: 10.13140/RG.2.1.4215.7846.

Massari, C., Brocca, L., Tarpanelli, A., Ciabatta, L., Camici, S., Moramarco, T., Giriraj, A., Dorigo, W. and Wagner, Wolfgang. (2015). 'Assessing the potential of CCI soil moisture product for data assimilation into rainfall-runoff modelling: a case study for the Niger River', Earth Observation for Water Cycle Science 2015, At ESA-ESRIN Frascati (Rome), Italy.

McDonnell, J. J. et al. (2001) 'Hydrology and biogeochemistry of forested catchments', Hydrological Processes, 15(10), pp. 1673–1674. doi: 10.1002/hyp.351.

Merheb, M. et al. (2016) 'Hydrological response characteristics of Mediterranean catchments at different time scales: a meta-analysis', Hydrological Sciences Journal, 61(14), pp. 2520–2539. doi: 10.1080/02626667.2016.1140174.

Merz, B. et al. (2010) 'Review article "assessment of economic flood damage"', Natural Hazards and Earth System Science, 10(8), pp. 1697–1724. doi: 10.5194/nhess-10-1697-2010.

Michailidi, E., S. et al. (2017) 'Adaptation of the concept of varying time of concentration within flood modelling: Theoretical and empirical investigations across the Mediterranean', in Geophysical Research Abstracts, Vol. 19. European Geosciences Union General Assembly 2017.

Michailidi, E. (2018) Flood risk assessment in gauged and ungauged basins in a multidimensional context. Universita Degli Studi di Brescia.

Moore, R. J. et al. (2006) ‘Issues in flood forecasting: Ungauged basins, extreme floods and uncertainty’, Frontiers in Flood Research / Le point de la recherche sur les crues, (305), pp. 103–122.

Paniconi, C. and Wood, E. F. (1993) ‘A detailed model for simulation of catchment scale subsurface hydrologic processes’, Water Resources Research, 29(6), pp. 1601–1620. doi: 10.1029/92WR02333.

Petroselli, A. and Grimaldi, S. (2018) ‘Design hydrograph estimation in small and fully ungauged basins: a preliminary assessment of the EBA4SUB framework’, Journal of Flood Risk Management, 11, pp. S197–S210. doi: 10.1111/jfr3.12193.

Pettyjohn, W. A. and Henning, R. (1979) ‘Preliminary Estimates of Ground-Wter Recharge Rates, Related Streamflow and Water Quality in Ohio’, American journal of public health and the nation’s health, p. 323. doi: 10.2105/AJPH.49.9.1190.

Pontikos, S. (2014) A probabilistic approach of soil moisture conditions in Greece. National Technical University of Athens.

Refsgaard, J. C. and Knudsen, J. (1996) ‘Operational Validation and Intercomparison of Different Types of Hydrological Models’, Wster Resources Research, 32(7), pp. 2189–2202.

Refsgaard, J. C. and Storm, B. (1995) ‘MIKE SHE’, in Computer Models of Watershed Hydrology, pp. 809–846.

Risva, K. et al. (2018) ‘A Framework for Dry Period Low Flow Forecasting in Mediterranean Streams’, Water Resources Management. doi: 10.1007/s11269-018-2060-z.

Rossetto, R., Borsi, I. and Foglia, L. (2015) ‘FREEWAT: FREE and open source software tools for WATer resource management’, Rendiconti online della Società Geologica Italiana, 35, pp. 252–255. doi: 10.3301/ROL.2015.113.

Savvidou, E. et al. (2018) ‘The Curve Number Concept as a Driver for Delineating Hydrological Response Units’, Water, 10(2), p. 194. doi: 10.3390/w10020194.

Sellinger, C. E. (1996) Computer program for performing hydrograph separation using the rating curve method. Michigan. Available at: https://permanent.access.gpo.gov/lps95098/glerl-100/tm-100.pdf.

Sloto, R. A. and Crouse, M. Y. (1996) HYSEP: A Computer Program for Streamflow Hydrograph Separation and Analysis. doi: 10.3133/wri964040.

Smith, M. et al. (2004) ‘The distributed model intercomparison project (DMIP): motivation and experiment design’, Journal ofHydrology, (298), pp. 4–26.

Soediono, B. (1989) ‘Think Python’, Journal of Chemical Information and Modeling, 53, p. 160. doi: 10.1017/CBO9781107415324.004.

Sorooshian, S. et al. (2008) Hydrological Modelling and the Water Cycle: Coupling the Atmospheric and Hydrological Models. Springer.

Su, N. (1995) 'The unit hydrograph model for hydrograph separation', Environment International, 21(5), pp. 509–515. doi: 10.1016/0160-4120(95)00050-U.

Szilagyi, J., Parlange, M. B. and Albertson, J. D. (1998) 'Recession flow analysis for aquifer parameter determination', Water Resources Research, 34(7), pp. 1851–1857. doi: 10.1029/98WR01009.

Tallaksen, L. M. (1995) 'A review of baseflow recession analysis', Journal of Hydrology, 165(1–4), pp. 349–370. doi: 10.1016/0022-1694(94)02540-R.

The National Regulator for Compulsory Specifications (NRCS). (2004) National Engineering Handbook: Part 630 -Hydrology. DC.

Tomer, S. K. (2011) 'Python in Hydrology', Book, pp. 1–157.

Uhlenbrook, S. and Hoeg, S. (2003) 'Quantifying uncertainties in tracer-based hydrograph separations: a case study for two-, three- and five-component hydrograph separations in a mountainous catchment', Hydrological Processes, 17(2), pp. 431–453. doi: 10.1002/hyp.1134.

Verma, S. et al. (2017a) 'A revisit of NRCS-CN inspired models coupled with RS and GIS for runoff estimation', Hydrological Sciences Journal. doi: 10.1080/02626667.2017.1334166.

Verma, S. et al. (2017b) 'A revisit of NRCS-CN inspired models coupled with RS and GIS for runoff estimation', Hydrological Sciences Journal, 62(12), pp. 1891–1930. doi: 10.1080/02626667.2017.1334166.

Ward, R. C. and Robinson, M. (1989) Principals of Hydrology. 3rd edn. London: McGraw-Hill.

Wickert, A. (2012) 'GRASS GIS for Geomorphologists: An Introductory Guide', Software Manual, pp. 1–67.

Wigmosta, M. S. and Lettenmaier, D. P. (1999) 'A comparison of simplified methods for routing topographically driven subsurface flow', Water Resources Research, 35(1), pp. 255–264. doi: 10.1029/1998WR900017.

Xu, Y. et al. (2011) 'Watershed discretization based on multiple factors and its application in the Chinese Loess Plateau', Hydrology and Earth System Sciences Discussions, 8(5), pp. 9063–9087. doi: 10.5194/hessd-8-9063-2011.