ΜΥΕ029 – Προσομοίωση και Μοντελοποίηση Υπολογιστικών Συστημάτων

2η Εργαστηριακή Άσκηση

Προσομοίωση ετερογενών συμπλεγμάτων εξυπηρετητών

(Παράδοση έως 30/5/2022)

1. Εισαγωγή

Οι εταιρίες παροχής υπηρεσιών διαδικτύου χρησιμοποιούν μια συλλογή από συμπλέγματα εξυπηρετητών για την αποδοτικότερη εξυπηρέτηση των πελατών τους. Τα σύγχρονα συμπλέγματα διακομιστών είναι συνήθως ετερογενή, δηλαδή αποτελούνται από εξυπηρετητές με διαφορετικές συνθέσεις. Το γεγονός αυτό οφείλεται κυρίως στη γρήγορη εξέλιξη που παρατηρείται στο υλικό των εξυπηρετητών μέσα σε μικρό χρονικό διάστημα και στη συνεχή αγορά και εγκατάσταση νέων εξυπηρετητών διαφορετικές χρονικές στιγμές. Ένα ενδιαφέρον ερευνητικό ερώτημα που προκύπτει είναι πως μπορεί να γίνει αποδοτικότερη η κατανομή των εργασιών των πελατών σε εξυπηρετητές με διαφορετικό ρυθμό επεξεργασίας έτσι ώστε η συνολική χρησιμοποίηση των πόρων να διατηρείται υψηλή.

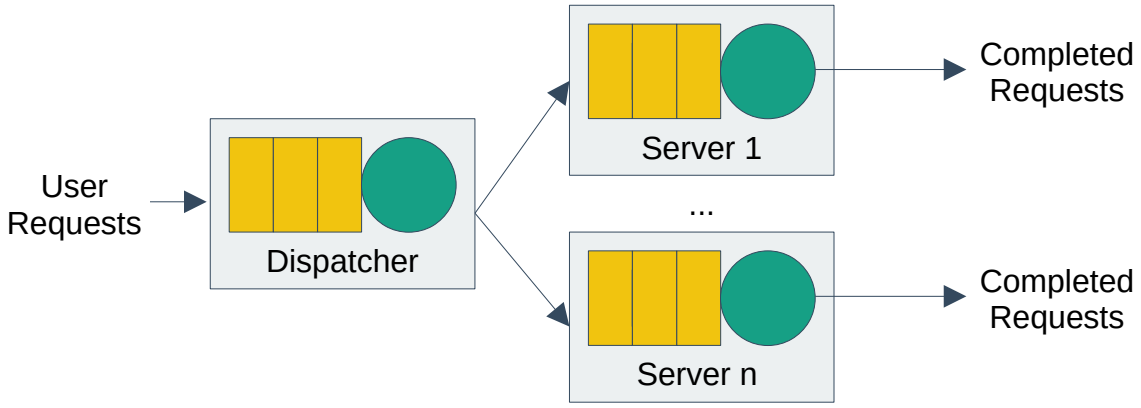
Στην παρούσα εργαστηριακή άσκηση θα χρησιμοποιήσετε τη μέθοδο της προσομοίωσης έτσι ώστε να προσπαθήσετε να απαντήσετε σε αυτό το ερώτημα. Συγκεκριμένα, θα μάθετε:

1. Να χρησιμοποιείτε τη μέθοδο της προσομοίωσης για να λύνετε σχεδιαστικά προβλήματα.
2. Να χρησιμοποιείτε έναν προσομοιωτή διακριτών γεγονότων για να προσομοιώνετε υπολογιστικά συστήματα.
3. Να χρησιμοποιείτε στατιστικά ορθές μεθόδους για την ανάλυση των αποτελεσμάτων της προσομοίωσης.

2. Εξισορρόπηση φορτίου σε συμπλέγματα διακομιστών

Ένα σύμπλεγμα διακομιστών αποτελείται από εκατοντάδες ή και χιλιάδες διακομιστές. Μια τυπική αρχιτεκτονική ενός συμπλέγματος διακρίνεται στην Εικόνα 1. Οι αιτήσεις των πελατών φτάνουν σε έναν διακομιστή που παίζει το ρόλο του διανομέα (dispatcher) και αναλαμβάνει να δρομολογήσει κάθε αίτηση σε έναν από πολλαπλούς διακομιστές εργάτες (workers) ώστε να την εξυπηρετήσει. Ο διανομέας προσπαθεί να διανέμει με έξυπνο τρόπο τις αιτήσεις των πελατών στους εργάτες έτσι ώστε

να διατηρεί όλους τους εξυπηρετητές ομοιόμορφα απασχολημένους. Αυτή η διεργασία ονομάζεται εξισορρόπηση φορτίου.



Εικόνα 1: Ένα τυπικό σύμπλεγμα διακομιστών

Έχουν προταθεί μια πληθώρα από αλγορίθμους εξισορρόπησης φορτίου στο παρελθόν. Ένας απλός αλγόριθμος είναι η εξυπηρέτηση εκ περιτροπής (Round-Robin). Υποθέτοντας ότι είναι διαθέσιμοι n εξυπηρετητές, ο αλγόριθμος λειτουργεί ως εξής: προωθεί την πρώτη αίτηση στον πρώτο εξυπηρετητή, τη δεύτερη στον δεύτερο εξυπηρετητή, την n-οστή αίτηση στον n-οστό εξυπηρετητή και στη συνέχεια ξεκινάει από την αρχή προωθώντας την n+1 αίτηση στον πρώτο εξυπηρετητή κοκ. Ένα πλεονέκτημα αυτής της μεθόδου είναι ότι ο διανομέας δε χρειάζεται να επικοινωνεί με τους εξυπηρετητές εργάτες, για παράδειγμα για να μάθει την τρέχουσα χρησιμοποίησή τους. Ωστόσο, αυτό το πλεονέκτημα αποτελεί και σοβαρό μειονέκτημα καθώς ο διανομέας δεν έχει γνώση για τη χρησιμοποίηση του κάθε εξυπηρετητή. Έτσι, είναι δυνατό ο διανομέας να προωθήσει μια αίτηση σε έναν εξυπηρετητή που είναι απασχολημένος, ενώ υπάρχουν άλλοι αδρανείς εξυπηρετητές.

Ένας αλγόριθμος εξισορρόπησης φορτίου που λαμβάνει υπόψιν την τρέχουσα χρησιμοποίηση των εξυπηρετητών είναι ο αλγόριθμος μικρότερου μήκους ουράς (shortest queue). Όταν ο διανομέας λαμβάνει μια αίτηση, επικοινωνεί με κάθε εξυπηρετητή για να μάθει τον αριθμό των εργασιών που βρίσκονται στην ουρά του. Μόλις λάβει αυτήν την πληροφορία από όλους τους εξυπηρετητές, ο διανομέας επιλέγει τον εξυπηρετητή με τη μικρότερη ουρά και προωθεί σε αυτόν την αίτηση.

Μια υπόθεση του αλγορίθμου μικρότερου μήκους ουράς είναι ότι όλοι οι εξυπηρετητές έχουν τον ίδιο ρυθμό επεξεργασίας. Ωστόσο τυπικά συμπλέγματα εξυπηρετητών πλέον αποτελούνται από εξυπηρετητές με ετερογενείς ρυθμούς επεξεργασίας. Μια δεύτερη υπόθεση είναι ότι ο διανομέας μπορεί να λάβει το μήκος ουράς κάθε εξυπηρετητή του συμπλέγματος. Ωστόσο κάτι τέτοιο δεν είναι πρακτικό σε συμπλέγματα με μεγάλο αριθμό από εξυπηρετητές. Μια ιδέα (δε θα χρειαστεί να την υλοποιήσετε στην παρούσα εργασία) είναι ο διανομέας να συλλέγει το μήκος της ουράς από ένα τυχαίο υποσύνολο των εξυπηρετητών.

Η σύνθεση του συμπλέγματος εξυπηρετητών το οποίο θα μελετήσετε παρουσιάζεται στον Πίνακα 1. Αποτελείται από έναν κόμβο διανομέα και 8 κόμβους εργάτες. Ο διανομέας έχει το αναγνωριστικό 1,

και οι κόμβοι εργάτες αριθμούνται από το 2 έως και το 9. Ο διανομέας δέχεται νέες αιτήσεις και τις προωθεί στους εξυπηρετητές εργάτες για εξυπηρέτηση. Κάθε κόμβος διαθέτει μια ουρά για εργασίες που αναμένουν εξυπηρέτηση με απεριόριστο αριθμό από θέσεις και μια μονάδα επεξεργασίας. Οι εξυπηρετητές 2-6 χρειάζονται κατά μέσο όρο 8 sec για να ολοκληρώσουν μια αίτηση, ενώ οι εξυπηρετητές 7-9 χρειάζονται κατά μέσο όρο 5 sec. Οι χρόνοι εξυπηρέτησης ακολουθούν εκθετική κατανομή. Νέες εργασίες φτάνουν στον διανομέα με μέσο χρόνο μεταξύ αφίξεων 4 sec και ακολουθούν εκθετική κατανομή.

Πίνακας 1: Σύνθεση και χαρακτηριστικά συμπλέγματος εξυπηρετητών

Αναγνωριστικό κόμβου	**Είδος**	**Μέσος χρόνος μεταξύ αφίξεων**	**Κατανομή αφίξεων**	**Μέσος χρόνος ολοκλήρωσης αίτησης**	**Κατανομή χρόνου εξυπηρέτησης**
1	Διανομέας	4 sec	Εκθετική	0	Ντετερμινιστική
2-6	Εργάτης	-		8 sec	Εκθετική
7-9	Εργάτης	-		5 sec	Εκθετική

3. Αλγόριθμοι εξισορρόπησης φορτίου

Κάθε φορά που λαμβάνει μια νέα αίτηση, ο διανομέας χρησιμοποιεί έναν αλγόριθμο εξισορρόπησης φορτίου ώστε να προωθήσει την αίτηση σε έναν εξυπηρετητή εργάτη. Ο Αλγόριθμος 1 που διακρίνεται στον Πίνακα 2 επιλέγει τον λιγότερο απασχολούμενο κόμβο χωρίς να λαμβάνει υπόψιν την ετερογένεια στον ρυθμό επεξεργασίας αιτήσεων.

Πίνακας 2: Ψευδοκώδικας για τον αλγόριθμο εξισορρόπησης φορτίου 1

```
Input: Nodes[]
Output: Node ID

min = Nodes[2]

for i in 3 to 9 do
    if Nodes[i].number_of_customers < min
        min = Nodes[i]
    endif
done
return min
```

Ο Αλγόριθμος 2 (Πίνακας 3) επιλέγει τον λιγότερο απασχολούμενο κόμβο λαμβάνοντας υπόψιν την ετερογένεια στο ρυθμό επεξεργασίας. Ο αλγόριθμος αρχικά ταξινομεί σε μια λίστα τους κόμβους ως προς το βαθμό απασχόλησής τους (αριθμός αιτήσεων). Στη συνέχεια, αν ο πρώτος κόμβος στην ταξινομημένη λίστα είναι γρήγορος, τότε επιστρέφει αμέσως αυτόν τον κόμβο. Διαφορετικά, εξετάζει τους επόμενους κόμβους μέχρι να βρει έναν γρήγορο κόμβο. Αν ο αριθμός εργασιών στο γρήγορο

κόμβο είναι έως και d παραπάνω από τις αντίστοιχες εργασίες του πρώτου κόμβου στη λίστα, τότε ο αλγόριθμος επιστρέφει τον γρήγορο κόμβο. Διαφορετικά, επιστρέφει τον πρώτο κόμβο στην ταξινομημένη λίστα.

Πίνακας 3: Ψευδοκώδικας για τον αλγόριθμο εξισορρόπησης φορτίου 2

```
Input: Nodes[]
Output: Node

for node_id in 2 to 9 do
    Busyness[node_id-2].size = Nodes[node_id].number_of_customers
    Busyness[node_id-2].id = node_id
done
sorted_nodes = sort(Busyness by size)
if sorted_nodes[0].id > 6 then              # 7-9: Fast node
    return Nodes[sorted_nodes[0].id]
else
    for i from 1 to 19 do
        if sorted_nodes[i].id > 6 then
            if sorted_nodes[i].size – d > sorted_nodes[0].size then
                return Nodes[sorted_nodes[i].id]
            else
                break
            endif
         endif
    done
    return Nodes[sorted_nodes[0].id]
endif
```

4. Ζητούμενα

Στην εργασία σας θα χρησιμοποιήσετε τη βιβλιοθήκη προσομοίωσης Ciw ώστε να αναπτύξετε έναν προσομοιωτή για το σύστημα που περιγράφεται στις ενότητες 2 και 3. Η Ciw είναι μια βιβλιοθήκη προσομοίωσης διακριτού γεγονότος για ανοικτά δίκτυα αναμονής. Για τις ανάγκες της άσκησης, θα χρειαστεί να μελετήσετε τον οδηγό χρήσης της βιβλιοθήκης Ciw, που θα βρείτε εδώ.

Το βασικό στοιχείο της Ciw είναι το αντικείμενο προσομοίωσης το οποίο αποτελείται από ένα δίκτυο κόμβων. Το αντικείμενο προσομοίωσης μπορεί να δημιουργηθεί με τη συνάρτηση ciw.create_network.

Το αντικείμενο προσομοίωσης αποτελείται από ένα δίκτυο κόμβων (nodes). Κάθε κόμβος περιλαμβάνει α) μια κατανομή που περιγράφει τους χρόνους μεταξύ αφίξεων νέων εργασιών, β) μια κατανομή που περιγράφει τους χρόνους εξυπηρέτησης, γ) έναν αριθμό από μονάδες επεξεργασίας. Στην περίπτωσή σας θεωρείστε ότι κάθε κόμβος αποτελείται από 1 μονάδα επεξεργασίας.

Η συνάρτηση ciw.seed() ορίζει ένα σπόρο για την αρχικοποίηση της γεννήτριας τυχαίων αριθμών. Είναι σημαντικό σε κάθε επανάληψη της προσομοίωσής σας να χρησιμοποιείτε διαφορετικό σπόρο.

Ο πίνακας δρομολόγησης (routing) καθορίζει τις πιθανότητες δρομολόγησης μιας αίτησης στους διάφορους κόμβους του δικτύου. Στη δική σας περίπτωση οι πιθανότητες αυτές θα είναι 0 μιας και ο αρχικός κόμβος (διανομέας) θα χρησιμοποιεί έναν από τους αλγορίθμους εξισορρόπησης φορτίου για να αναδιανέμει τις αιτήσεις στους υπόλοιπους κόμβους.

4.1. Περιβάλλον

Για την εκτέλεση της προσομοίωσης σας δίνεται μια εικονική μηχανή VMWare που έχει εγκατεστημένο το Λειτουργικό Σύστημα Debian 11. Για να εκτελέσετε την εικονική μηχανή, θα χρειαστείτε το σύστημα εικονικοποίησης VMWare Workstation Player, το οποίο μπορείτε να το κατεβάσετε δωρεάν για περιβάλλον Windows ή Linux από τον σύνδεσμο:

https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

Αφού κατεβάσετε και εγκαταστήσετε το Player, μπορείτε να κατεβάσετε την εικονική μηχανή από τον παρακάτω σύνδεσμο, αφού πρώτα συνδεθείτε στον ιδρυματικό σας λογαριασμό:

https://drive.google.com/file/d/1yfdNzQMLYyb1Cu033BeNZWf1AMGTET8e/view?usp=sharing

Αν έχετε ήδη κατεβάσει την εικονική μηχανή από την πρώτη άσκηση, μπορείτε να εγκαταστήσετε το ciw ως εξής:

$ pip install ciw

Εναλλακτικά, μπορείτε να εγκαταστήσετε με την παραπάνω διαδικασία το ciw στο δικό σας σύστημα και να το χρησιμοποιήσετε για τις ανάγκες της άσκησης.

4.2. Είσοδος προσομοίωσης

Ο προσομοιωτής που θα υλοποιήσετε θα δέχεται ως είσοδο το είδος του αλγορίθμου εξισορρόπησης φορτίου (load_balancing), την παράμετρο d, και τη διάρκεια της προσομοίωσης σε δευτερόλεπτα (sim_time_secs). Επιπλέον, θα δέχεται τα χαρακτηριστικά του συμπλέγματος εξυπηρετητών, όπως περιγράφονται στον Πίνακα 1.

Μπορείτε να ορίσετε τις παραμέτρους εισόδου που αφορούν το σύστημα σε ένα αρχείο με κατάληξη .yml και στη συνέχεια να χρησιμοποιήσετε τη μέθοδο ciw.create_network_from_yml() ώστε να φορτώσετε το αρχείο στον προσομοιωτή. Τις υπόλοιπες παραμέτρους μπορείτε να τις διαβάζετε από τη γραμμή εκτέλεσης (1η παράμετρος: *διάρκεια προσομοίωσης*, 2η παράμετρος: *αλγόριθμος εξισορρόπησης*, 3η παράμετρος: *d*).

4.3. Διαδικασία προσομοίωσης

Υλοποιήστε ένα δίκτυο ουρών που αναπαριστά το σύστημα που φαίνεται στην Εικόνα 1 και τον Πίνακα 1. Στην ενότητα 4.20 του οδηγού (σελ. 54) μπορείτε να δείτε ένα παράδειγμα παρόμοιου συστήματος. Εσείς αγνοήστε την υπόθεση της κοινοχρησίας που αναφέρει το παράδειγμα. Υλοποιήστε δύο συναρτήσεις RoutingDecision1 και RoutingDecision2 για κάθε έναν από τους δύο αλγορίθμους εξισορρόπησης φορτίου. Ο προσομοιωτής θα επιλέγει τον αλγόριθμο ανάλογα με την είσοδο που λαμβάνει από το χρήστη.

Η προσομοίωση του συστήματος θα εκτελείται για χρόνο ίσο με 24 ώρες. Ωστόσο, επειδή η αρχική κατάσταση του συστήματος (άδειες ουρές σε όλους τους εξυπηρετητές) είναι μη ρεαλιστική, θα χρειαστεί να συμπεριλάβετε έναν επιπλέον χρόνο ζεστάματος (warm-up time) ώστε το σύστημα να εισέλθει σε μια κατάλληλη κατάσταση για συλλογή αποτελεσμάτων (1 ώρα).

Είναι σημαντικό να εκτελέσετε πολλαπλές επαναλήψεις της προσομοίωσης έτσι ώστε τα αποτελέσματα που θα λάβετε να είναι στατιστικά ορθά.

Χρησιμοποιώντας τη μέθοδο των ανεξάρτητων επαναλήψεων, δημιουργήστε τα διαστήματα εμπιστοσύνης για το μέσο χρόνο αναμονής με επίπεδο εμπιστοσύνης 95% (α=0.05) και σχετικό σφάλμα 5% (γ=0.05). Θεωρείστε ότι κάθε επανάληψη διαρκεί για sim_time_secs δευτερόλεπτα. Χρησιμοποιήστε τον πίνακα με τους αριθμούς t n-1,1-α/2 που σας δίνεται για τον υπολογισμό των διαστημάτων εμπιστοσύνης.

4.4. Έξοδος προσομοίωσης

Ο προσομοιωτής σας θα υπολογίζει τις ακόλουθες μετρικές: **α) μέσο χρόνο αναμονής, β) μέσο ρυθμό εξυπηρέτησης αιτήσεων. γ) μέση συνολική χρησιμοποίηση κόμβων, δ) μέση χρησιμοποίηση ανά κόμβο** (ξεχωριστά για κάθε έναν από τους 9 εξυπηρετητές εργάτες).

Επιπλέον, θα δημιουργεί ένα αρχείο εξόδου στο δίσκο στο οποίο θα παρουσιάζει με ορθό τρόπο τα αποτελέσματα της προσομοίωσης. Το αρχείο θα πρέπει να χωρίζεται σε τρεις ενότητες: **α) σύντομη περιγραφή του συστήματος, β) παράμετροι εισόδου, γ) αποτελέσματα εξόδου.**

Μπορείτε επίσης να χρησιμοποιήσετε διάφορες βιβλιοθήκες της Python για να δημιουργήσετε γραφικές παραστάσεις των αποτελεσμάτων (μετρικές: β, γ, και δ). Οι γραφικές παραστάσεις θα πρέπει να ακολουθούν τη μορφή που διδαχτήκατε στην αντίστοιχη διάλεξη του μαθήματος, ώστε να παρουσιάζουν τα αποτελέσματα με σαφήνεια.

Μπορείτε να ορίσετε τις παραμέτρους εισόδου του προσομοιωτή σε ένα αρχείο με κατάληξη .yml και στη συνέχεια να χρησιμοποιήσετε τη μέθοδο ciw.create_network_from_yml() ώστε να φορτώσετε το αρχείο στον προσομοιωτή.

5. Παράδοση

Αφού υλοποιήσετε τον προσομοιωτή, το επόμενο βήμα είναι να τον χρησιμοποιήσετε ώστε να απαντήσετε στα επόμενα ερωτήματα:

1. Για τον Αλγόριθμο εξισορρόπησης φορτίου 2, ποια τιμή του d δίνει τον μικρότερο μέσο χρόνο αναμονής; Ποια τιμή του d δίνει την υψηλότερη μέση χρησιμοποίηση;
2. Συγκρίνοντας στατιστικά τους Αλγορίθμους 1 και 2, μπορεί ο Αλγόριθμος 2 να πετύχει χαμηλότερο μέσο χρόνο αναμονής και υψηλότερη χρησιμοποίηση των εξυπηρετητών από τον Αλγόριθμο 1;

Θα χρειαστεί επίσης να δημιουργήσετε ένα σενάριο (script) που θα αυτοματοποιεί την εκτέλεση του προσομοιωτή για όλες τις περιπτώσεις που θα μελετήσετε. Προσέξτε ότι για την επιλογή του αριθμού επαναλήψεων σε κάθε περίπτωση θα είναι υπεύθυνος ο προσομοιωτής και όχι το σενάριο αυτοματοποίησης.

Τέλος, γράψτε μια αναφορά η οποία θα περιλαμβάνει τα εξής:

1. Λεπτομερής περιγραφή του προσομοιωτή.
2. Οδηγίες εκτέλεσης του προσομοιωτή και των πειραμάτων.
3. Περιγραφή των αποτελεσμάτων, η οποία θα περιλαμβάνει τις γραφικές παραστάσεις που δημιουργήσατε και κείμενο το οποίο θα εξηγεί τα αποτελέσματα και θα απαντάει στα 2 ερωτήματα.
4. Τα συμπεράσματα της μελέτης.

Θα πρέπει να παραδώσετε τον κώδικα του προσομοιωτή σας, το αρχείο *report.pdf* που περιλαμβάνει την πλήρη αναφορά, το script που δημιουργήσατε για να αυτοματοποιήσετε τη διαδικασία αξιολόγησης, το αρχείο εισόδου στον προσομοιωτή, καθώς και κάθε αρχείο εξόδου που παρήγαγε ο προσομοιωτής σας. Δημιουργήστε ένα αρχείο simulation.zip που θα περιλαμβάνει όλα τα παραπάνω αρχεία.

Μπορείτε να παραδώσετε την εργασία είτε ατομικά, είτε σε ομάδες των δύο. Μη ξεχάσετε να συμπεριλάβετε στην πρώτη σελίδα του αρχείου *report.pdf* τα ονόματά και τους αριθμούς μητρώου και των δύο μελών της ομάδας.

Υποβάλετε τη λύση σας με την εντολή:

$ turnin lab2_22@mye029 simulation.zip

Οδηγίες για τη χρήση του turnin μπορείτε να βρείτε στον ακόλουθο σύνδεσμο:

https://support.cs.uoi.gr/doku.php?id=cse:%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CF%84%CE%AE%CF%81%CE%B9%CE%B1:turnin

6. Bonus 10%

Αντί για τη μέθοδο του ζεστάματος χρησιμοποιήστε τη μέθοδο διαγραφής αρχικών αποτελεσμάτων για να υπολογίσετε το μήκος της μεταβατικής κατάστασης. Αναφέρετε στην αναφορά σας το μήκος που βρήκατε και υπολογίστε όλα τα ζητούμενα στατιστικά από τα αποτελέσματα της σταθερής κατάστασης, έχοντας αφαιρέσει εκείνα της μεταβατικής περιόδου.

Παράρτημα 1

TABLE T.1
Critical points $t_{\nu,\gamma}$ for the t distribution with ν df, and z_γ for the standard normal distribution
$\gamma = P(T_\nu \le t_{\nu,\gamma})$, where T_ν is a random variable having the t distribution with ν df; the last row, where $\nu = \infty$, gives the normal critical points satisfying $\gamma = P(Z \le z_\gamma)$, where Z is a standard normal random variable

	γ												
ν	0.6000	0.7000	0.8000	0.9000	0.9333	0.9500	0.9600	0.9667	0.9750	0.9800	0.9833	0.9875	0.9900	0.9917	0.9938	0.9950
1	0.325	0.727	1.376	3.078	4.702	6.314	7.916	9.524	12.706	15.895	19.043	25.452	31.821	38.342	51.334	63.657
2	0.289	0.617	1.061	1.886	2.456	2.920	3.320	3.679	4.303	4.849	5.334	6.205	6.965	7.665	8.897	9.925
3	0.277	0.584	0.978	1.638	2.045	2.353	2.605	2.823	3.182	3.482	3.738	4.177	4.541	4.864	5.408	5.841
4	0.271	0.569	0.941	1.533	1.879	2.132	2.333	2.502	2.776	2.999	3.184	3.495	3.747	3.966	4.325	4.604
5	0.267	0.559	0.920	1.476	1.790	2.015	2.191	2.337	2.571	2.757	2.910	3.163	3.365	3.538	3.818	4.032
6	0.265	0.553	0.906	1.440	1.735	1.943	2.104	2.237	2.447	2.612	2.748	2.969	3.143	3.291	3.528	3.707
7	0.263	0.549	0.896	1.415	1.698	1.895	2.046	2.170	2.365	2.517	2.640	2.841	2.998	3.130	3.341	3.499
8	0.262	0.546	0.889	1.397	1.670	1.860	2.004	2.122	2.306	2.449	2.565	2.752	2.896	3.018	3.211	3.355
9	0.261	0.543	0.883	1.383	1.650	1.833	1.973	2.086	2.262	2.398	2.508	2.685	2.821	2.936	3.116	3.250
10	0.260	0.542	0.879	1.372	1.634	1.812	1.948	2.058	2.228	2.359	2.465	2.634	2.764	2.872	3.043	3.169
11	0.260	0.540	0.876	1.363	1.621	1.796	1.928	2.036	2.201	2.328	2.430	2.593	2.718	2.822	2.985	3.106
12	0.259	0.539	0.873	1.356	1.610	1.782	1.912	2.017	2.179	2.303	2.402	2.560	2.681	2.782	2.939	3.055
13	0.259	0.538	0.870	1.350	1.601	1.771	1.899	2.002	2.160	2.282	2.379	2.533	2.650	2.748	2.900	3.012
14	0.258	0.537	0.868	1.345	1.593	1.761	1.887	1.989	2.145	2.264	2.359	2.510	2.624	2.720	2.868	2.977
15	0.258	0.536	0.866	1.341	1.587	1.753	1.878	1.978	2.131	2.249	2.342	2.490	2.602	2.696	2.841	2.947
16	0.258	0.535	0.865	1.337	1.581	1.746	1.869	1.968	2.120	2.235	2.327	2.473	2.583	2.675	2.817	2.921
17	0.257	0.534	0.863	1.333	1.576	1.740	1.862	1.960	2.110	2.224	2.315	2.458	2.567	2.657	2.796	2.898
18	0.257	0.534	0.862	1.330	1.572	1.734	1.855	1.953	2.101	2.214	2.303	2.445	2.552	2.641	2.778	2.878
19	0.257	0.533	0.861	1.328	1.568	1.729	1.850	1.946	2.093	2.205	2.293	2.433	2.539	2.627	2.762	2.861
20	0.257	0.533	0.860	1.325	1.564	1.725	1.844	1.940	2.086	2.197	2.285	2.423	2.528	2.614	2.748	2.845
21	0.257	0.532	0.859	1.323	1.561	1.721	1.840	1.935	2.080	2.189	2.277	2.414	2.518	2.603	2.735	2.831
22	0.256	0.532	0.858	1.321	1.558	1.717	1.835	1.930	2.074	2.183	2.269	2.405	2.508	2.593	2.724	2.819
23	0.256	0.532	0.858	1.319	1.556	1.714	1.832	1.926	2.069	2.177	2.263	2.398	2.500	2.584	2.713	2.807
24	0.256	0.531	0.857	1.318	1.553	1.711	1.828	1.922	2.064	2.172	2.257	2.391	2.492	2.575	2.704	2.797
25	0.256	0.531	0.856	1.316	1.551	1.708	1.825	1.918	2.060	2.167	2.251	2.385	2.485	2.568	2.695	2.787
26	0.256	0.531	0.856	1.315	1.549	1.706	1.822	1.915	2.056	2.162	2.246	2.379	2.479	2.561	2.687	2.779
27	0.256	0.531	0.855	1.314	1.547	1.703	1.819	1.912	2.052	2.158	2.242	2.373	2.473	2.554	2.680	2.771
28	0.256	0.530	0.855	1.313	1.546	1.701	1.817	1.909	2.048	2.154	2.237	2.368	2.467	2.548	2.673	2.763
29	0.256	0.530	0.854	1.311	1.544	1.699	1.814	1.906	2.045	2.150	2.233	2.364	2.462	2.543	2.667	2.756
30	0.256	0.530	0.854	1.310	1.543	1.697	1.812	1.904	2.042	2.147	2.230	2.360	2.457	2.537	2.661	2.750
40	0.255	0.529	0.851	1.303	1.532	1.684	1.796	1.886	2.021	2.123	2.203	2.329	2.423	2.501	2.619	2.704
50	0.255	0.528	0.849	1.299	1.526	1.676	1.787	1.875	2.009	2.109	2.188	2.311	2.403	2.479	2.594	2.678
75	0.254	0.527	0.846	1.293	1.517	1.665	1.775	1.861	1.992	2.090	2.167	2.287	2.377	2.450	2.562	2.643
100	0.254	0.526	0.845	1.290	1.513	1.660	1.769	1.855	1.984	2.081	2.157	2.276	2.364	2.436	2.547	2.626
∞	0.253	0.524	0.842	1.282	1.501	1.645	1.751	1.834	1.960	2.054	2.127	2.241	2.326	2.395	2.501	2.576