

Τίτλος διπλωματικής

Διπλωματική εργασία
Ονοματεπώνυμο φοιτητή

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Όνομα εργαστηρίου
Πολυτεχνική Σχολή

Επιβλέπων:
Ημερομηνία

Thesis title

Diploma Thesis

Student name

Department of Electrical and Computer Engineering

Laboratory name

Faculty of Engineering

Advisor:

Date

Copyright © 2025 Ονοματεπώνυμο φοιτητή, Επιβλέπων καθηγητής

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευτεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Δυτικής Μακεδονίας.

Copyright © 2025 Student Name, Supervisor name

All rights reserved.

Copying, storing, and distributing this thesis, in whole or in part, for commercial purposes is prohibited. Reproduction, storage, and distribution for non-profit, educational, or research purposes are permitted, provided that the source is acknowledged and this message is preserved. Any inquiries regarding the use of this work for commercial purposes should be directed to the author.

The opinions and conclusions contained in this document express the views of the author and should not be interpreted as representing the official positions of the University of Western Macedonia.

Περίληψη

Κείμενο περίληψης

Λέξεις κλειδιά:

Abstract

Abstract text

Keywords:

Ευχαριστίες

Ευχαριστίες.

Acknowledgements

Your acknowledgements text is here.

Κατάλογος Πινάκων

2.2.1 Σύντομη περιγραφή	12
-----------------------------------	----

Κατάλογος Σχημάτων

2.2.1 Σύντομη περιγραφή	11
-----------------------------------	----

Κατάλογος Εικόνων

1.2.1 Σύντομη περιγραφή	9
2.2.1 Σύντομη περιγραφή	11

Κατάλογος Αλγορίθμων

2.1	Όνομα αλγορίθμου - Algorithm name	10
-----	---	----

Περιεχόμενα

1	Τίτλος κεφαλαίου	9
1.1	Τίτλος	9
1.2	Τίτλος	9
2	Τίτλος κεφαλαίου	10
2.1	Τίτλος	10
2.2	Τίτλος	11
	Παράρτημα Α': Κώδικας	13
	Βιβλιογραφία	18

Ακρωνύμια

AI Artificial Intelligence. 8

CPU Central Processing Unit. 8, 9

LP Linear Programming. 8

Κεφάλαιο 1

Τίτλος κεφαλαίου

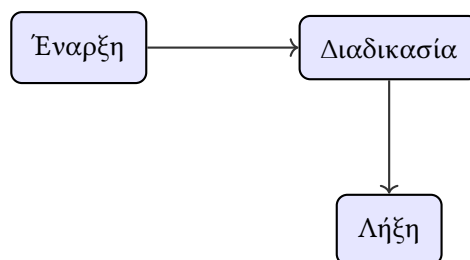
1.1 Τίτλος

Θεώρημα 1.1 (Όνομα θεωρήματος). *Περιγραφή*

Δοκιμαστικό κείμενο Central Processing Unit (*CPU*).

Λήμμα 1.2. Για κάθε $x \in \mathbb{R}$, ισχύει $x^2 \geq 0$.

1.2 Τίτλος



Εικόνα 1.2.1: Σύντομη περιγραφή

Κεφάλαιο 2

Τίτλος κεφαλαίου

2.1 Τίτλος

Παράδειγμα αναφοράς: [1]. Παράδειγμα αναφοράς: [2].

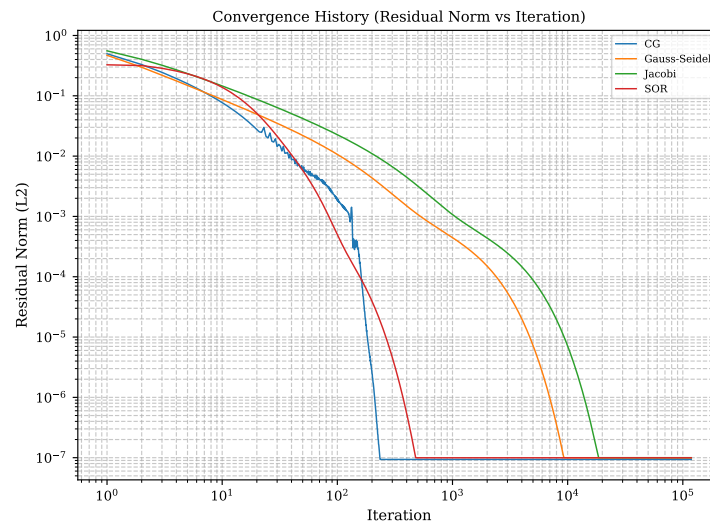
Αλγόριθμος 2.1 – Όνομα αλγορίθμου - Algorithm name

Require: Implication graph and a conflict at the current decision level

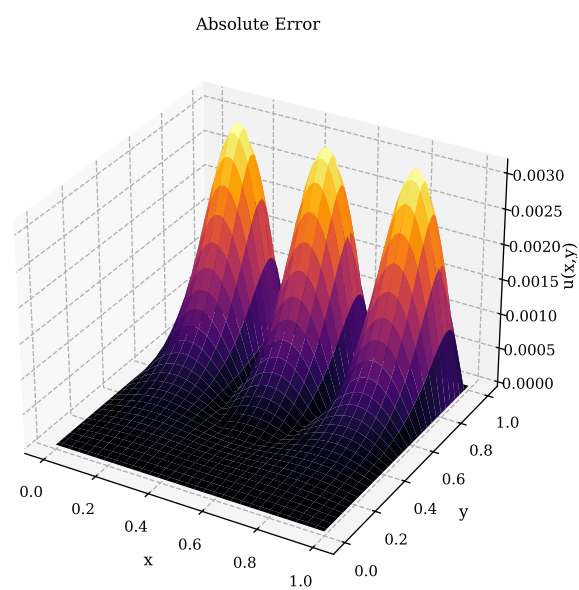
Ensure: Learned clause and backjump level

```
1: function ANALYZE_CONFLICT
2:   if current_decision_level() == 0 then
3:     return -1
4:   end if
5:    $cl \leftarrow \text{find\_conflicting\_clause}()$ 
6:   while not stop_criterion_met( $cl$ ) do
7:      $lit \leftarrow \text{choose\_literal}(cl)$ 
8:      $var \leftarrow \text{variable\_of\_literal}(lit)$ 
9:      $ante \leftarrow \text{antecedent}(var)$ 
10:     $cl \leftarrow \text{resolve}(cl, ante, var)$ 
11:  end while
12:  add_clause_to_database( $cl$ )
13:   $back\_dl \leftarrow \text{clause\_asserting\_level}(cl)$ 
14:  return  $back\_dl$ 
15: end function
```

2.2 Τίτλος



Σχήμα 2.2.1: Σύντομη περιγραφή



Εικόνα 2.2.1: Σύντομη περιγραφή

N	CG	Gauss-Seidel	Jacobi	SOR
0	8	14	64	35
1	18	39	324	62
2	28	64	784	87
3	38	90	1444	112
4	48	114	2304	136
5	58	139	3364	161
6	68	163	4624	225
7	78	187	6084	304
8	88	211	7614	388
9	98	235	9260	479
10	108	258	11048	577
11	118	282	12977	682
12	128	305	15043	794
13	138	328	17244	913
14	148	351	19577	1039
15	158	374	22041	1172
16	168	398	24633	1311
17	178	421	27352	1457
18	188	444	30196	1609
19	198	467	33163	1768

Πίνακας 2.2.1: Σύντομη περιγραφή

Παράρτημα Α΄

Κώδικας

```
1  def hello(name):
2      print("Hello", name)
```

```
1  #ifndef SOR_HPP
2  #define SOR_HPP
3
4  #include "utils/SolverLog.hpp"
5  #include "solvers/config.h"
6  template<typename Vector>
7  struct SOR
8  {
9      using Scalar      = typename Vector::Scalar;
10     using SparseMatrix = Eigen::SparseMatrix<Scalar, Eigen::RowMajor>;
11
12     double            tol      = DEFAULT_TOL;
13     int               max_iters = MAX_ITERS;
14     double            omega;
15     std::string       name      = "SOR";
16     SolverLog<Vector> log;
17     Vector             final_solution;
18
19     template<typename System>
```

```

20 SOR (System system) : omega(system.omega_)
21 {
22     log.system_dim      = system.A.rows();
23     max_iters           = static_cast<int>(10 *
24         ↪ std::sqrt(log.system_dim));
25     log.max_iterations = max_iters;
26     log.tolerance      = tol;
27     log.solver_name    = name;
28 }
29
30 template<typename System>
31 void solve(System& system)
32 {
33     const auto& A      = system.A;
34     const auto& b      = system.b;
35     auto& u            = system.u;
36
37     std::cout << "max_iters: " << max_iters << '\n';
38
39     double sum1, sum2;
40
41     double b_norm      = b.norm();
42     double res          = (A * u - b).norm() / b_norm;
43
44     if (res <= tol)
45     {
46         this->final_solution = u;
47         log.final_solution   = this->final_solution;
48         log.converged        = 1;
49         return;
50     }
51
52     Vector inv_diag = A.diagonal().cwiseInverse();
53
54     for (int k = 0; k < max_iters; k++)
55     {
56         for (int i = 0; i < A.rows(); ++i)
57         {
58             double sum = 0;

```

```

58
59     for (typename SparseMatrix::InnerIterator it(A, i);
        ↪ it; ++it)
60     {
61         int j = it.col();
62
63         if (j != i)
64         {
65             sum += it.value() * u[j];
66         }
67     }
68     u[i] = (1 - omega) * u[i] + omega * (inv_diag[i] * (b[i]
        ↪ - sum));
69 }
70
71 res = (A * u - b).norm() / b.norm();
72
73 log.num_of_iterations++;
74 log.res_per_iteration.push_back(res);
75
76 if (res <= tol)
77 {
78     this->final_solution = u;
79     log.final_solution    = this->final_solution;
80     log.converged = 1;
81     return;
82 }
83 }
84 this->final_solution = u;
85 log.final_solution    = this->final_solution;
86 return;
87 }
88 };
89
90
91 #endif // SOR_HPP

```

```

1  #ifndef CONJUGATE_GRADIENT_HPP
2  #define CONJUGATE_GRADIENT_HPP
3
4  #include "utils/SolverLog.hpp"
5  #include "solvers/config.h"
6  template< typename Vector>
7  struct ConjugateGradient
8  {
9      double          tol          = DEFAULT_TOL;
10     int              max_iters    = 1e6;
11     std::string      name         = "CG";
12     SolverLog<Vector> log;
13     Vector           final_solution;
14
15     ConjugateGradient ()
16     {
17         log.tolerance          = tol;
18         log.max_iterations     = max_iters;
19         log.solver_name        = name;
20     }
21
22     template<typename System>
23     void solve(System& system)
24     {
25         const auto& A = system.A;
26         const auto& b = system.b;
27         auto& u = system.u;
28         log.system_dim = A.rows();
29
30         std::cout << "max_iters: " << max_iters << '\n';
31
32         Vector r          = b - A * u; // initial residual
33         double b_norm = b.norm();
34         double r_norm = r.norm();
35
36         if (r_norm / b_norm <= tol)
37         {
38             this->final_solution = u;

```

```

39         log.final_solution    = this->final_solution;
40         log.converged          = 1;
41         return;
42     }
43
44     Vector d = r; // initial search direction
45     Vector Ad(A.rows());
46
47     for (int k = 0; k < max_iters; k++)
48     {
49         // std::cout << "----- iter. " << k+1 << "
50         ↪ -----\n";
51         Ad.noalias() = A * d;
52
53         double alpha      = ((r.transpose() * r) / (d.transpose() *
54         ↪ Ad)).coeff(0); // step size
55         double r_prev_dot = (r.transpose() * r).coeff(0); // to
56         ↪ calculate beta
57
58         u.noalias() += alpha * d;
59         r.noalias() -= alpha * Ad;
60
61         r_norm = r.norm();
62
63         log.num_of_iterations++;
64         log.res_per_iteration.push_back(r_norm / b_norm);
65
66         if (r_norm / b_norm <= tol)
67         {
68             log.converged          = 1;
69             this->final_solution = u;
70             log.final_solution    = this->final_solution;
71             return;
72         }
73
74         double beta = r.dot(r) / r_prev_dot;
75         d          = r + beta * d; // update direction
76     }
77     this->final_solution = u;

```

```
75         log.final_solution    = this->final_solution;  
76         return;  
77     }  
78 };  
79  
80  
81 #endif // CONJUGATE_GRADIENT_HPP
```

Βιβλιογραφία

- [1] V. Balabanov and J.-H. R. Jiang, “Unified qbf certification and its applications,” *Form. Methods Syst. Des.*, vol. 41, p. 45–65, Aug. 2012.
- [2] Γιάννης Παπαδόπουλος, *Μαθηματικά για Επιστήμονες Υπολογιστών*. Εκδόσεις Πανεπιστημίου Μακεδονίας, 2020.