Παράδειγμα #4
ΕΠΙΛΥΣΗ ΓΡΑΜΜΙΚΩΝ ΑΛΓΕΒΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΜΕΘΟΔΟΥΣ
ΕΠΙΜΕΛΕΙΑ: Ν. Βασιλειάδης

Άσκηση 1

Τα ισοζύγια μάζας του συστήματος διανομή ατμού σε μονάδα διυλιστηρίου δίνονται από τις παρακάτω εξισώσεις:

$$181.60 - x_3 - 132.57 - x_4 - x_5 = 5.1$$
$$1.17x_3 - x_6 = 0$$
$$132.57 - 0.745x_7 = 61.2$$
$$x_5 + x_7 - x_8 - x_9 - x_{10} + x_2 = 99.1$$
$$x_8 + x_9 + x_{10} + x_{11} - x_{12} - x_{13} = -8.4$$
$$x_6 - x_2 = 24.2$$
$$-1.15(181.60) + x_3 - x_6 + x_{12} + x_1 = -19.7$$
$$181.60 - 4.594x_{12} - 0.11x_1 = 35.05$$
$$-0.0423(181.60) + x_{11} = 2.88$$
$$-0.016(181.60) + x_4 = 0$$
$$x_8 - 0.0147x_1 = 0$$
$$x_5 - 0.07x_{14} = 0$$
$$-0.0805(181.60) + x_9 = 0$$
$$x_{12} - x_{14} + x_1 = -97.9$$

Τα x έχουν μονάδες σε 1000 kg/hr. Τα ισοζύγια μάζας εφαρμόζονται στις συσκευές: Header 680-psia, De-super heater, Alternator turbine, Header 170-psia, Header 37-psia, Header 215-psia, BFW balance, Condensate drum, Blow down flash drum, Boiler atomizing, Treated feed-water pump, Boiler feed-water pump, Boiler fan, De-aerator.

Να προσδιοριστούν οι 14 άγνωστοι, λύνοντας το γραμμικό σύστημα με τις επαναληπτικές μεθόδους α) Jacobi με β) Gauss-Seidel και γ) S.O.R. (να γίνει παραμετρική μελέτη του απαιτούμενου αριθμού επαναλήψεων σε σχέση με το ω).

Σε όλες τις μεθόδους να χρησιμοποιηθεί κριτήριο τερματισμού: $\varepsilon = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(x_i^{(m+1)} - x_i^{(m)}\right)^2} < 10^{-5}$

Απάντηση:
Είναι φανερό πως για τις μεθόδους Jacobi, Gauss-Seidel και S.O.R είναι αναγκαίο να αναδιαταχθούν οι εξισώσεις ώστε τα διαγώνια στοιχεία του πίνακα συντελεστών A να είναι μη μηδενικά. Εκτελώντας τις πράξεις και αναδιατάσσοντας τις εξισώσεις προκύπτει το ακόλουθο σύστημα (**σημειώνεται ότι ο νέος πίνακας συντελεστών έχει μη μηδενικά διαγώνια στοιχεία αλλά δεν είναι διαγώνια υπέρτερος**!!!!):

$$x_1 + x_3 - x_6 + x_{12} = 189.14$$
$$-x_2 + x_6 = 24.2$$
$$x_3 + x_4 + x_5 = 43.93$$
$$x_4 = 2.9056$$
$$x_5 - 0.07x_{14} = 0$$
$$1.17x_3 - x_6 = 0$$
$$0.745x_7 = 71.37$$
$$-0.0147x_1 + x_8 = 0$$
$$x_9 = 14.6188$$
$$x_2 + x_5 + x_7 - x_8 - x_9 - x_{10} = 99.1$$
$$x_{11} = 10.5617$$
$$0.11x_1 + 4.594x_{12} = 146.55$$
$$x_8 + x_9 + x_{10} + x_{11} - x_{12} - x_{13} = -8.4$$
$$x_1 + x_{12} - x_{14} = -97.9$$

Άρα το παραπάνω γραμμικό σύστημα μπορεί να γραφεί στην μορφή $\boldsymbol{Ax = b}$, όπου:

$\boldsymbol{A} =$

1	0	1	0	0	-1	0	0	0	0	0	1	0	0
0	-1	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	-0.07
0	0	1.17	0	0	-1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.745	0	0	0	0	0	0	0
-0.0147	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	1	0	1	-1	-1	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0
0.11	0	0	0	0	0	0	0	0	0	0	4.594	0	0
0	0	0	0	0	0	0	1	1	1	1	-1	-1	0
1	0	0	0	0	0	0	0	0	0	0	1	0	-1

$\boldsymbol{b} =$

189.14
24.2
43.93
2.9056
0
0
71.37
0
14.6188
99.1
10.5617
146.55
-8.4
-97.9

Όπως δείχθηκε στο Παράδειγμα 3 με την βοήθεια της Mathematica και χρήση της εντολής LinearSolve η λύση του γραμμικού συστήματος προκύπτει:

```
{164.7,0.00196857,20.6854,2.9056,20.339,24.202,95.7987,
2.42109,14.6188,-0.000304565,10.5617,27.9567,8.0446,290.557}
```

1α. Jacobi

Πρόγραμμα σε Fortran για την μέθοδο Jacobi:

```
program Jacobi
implicit none
integer::i,j,iter,maxiter,n
real*8:: s,rel,err,tstart,tend
real*8,allocatable:: a(:,:),xold(:),xnew(:)

!Find program start time
call cpu_time(tstart)

!Open output file
open (100,file="Jacobi_results.dat")

!Definition of parameters for the Jacobi method
n=14                                    !Number of equations
allocate(a(n,n+1),xold(n),xnew(n))      !Allocation of required matrices
maxiter=10000                           !Maximum number of iteration
rel=1d-5                                !Desired relative error
xold=0.                                 !Initial guess x0
!Definition of the linear system of equations
a(1,:)=(/1.,0.,1.,0.,0.,-1.,0.,0.,0.,0.,0.,1.,0.,0.,189.14/)
a(2,:)=(/0.,-1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,24.2/)
a(3,:)=(/0.,0.,1.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,43.93/)
a(4,:)=(/0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,2.9056/)
a(5,:)=(/0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,-0.07,0./)
a(6,:)=(/0.,0.,1.17,0.,0.,-1.,0.,0.,0.,0.,0.,0.,0.,0./)
a(7,:)=(/0.,0.,0.,0.,0.,0.,0.745,0.,0.,0.,0.,0.,0.,0.,71.37/)
a(8,:)=(/-0.0147,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0./)
a(9,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,14.6188/)
a(10,:)=(/0.,1.,0.,0.,1.,0.,1.,-1.,-1.,-1.,0.,0.,0.,0.,99.1/)
a(11,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,10.5617/)
a(12,:)=(/0.11,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,4.594,0.,0.,146.55/)
a(13,:)=(/0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.,-1.,-1.,0.,-8.4/)
a(14,:)=(/1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,-1.,-97.9/)

!Write system of equations to be solved to output file
write(100,"(A)") "-----------------------------------------------"
write(100,"(A)") "--------- Linear system of equations ----------"
write(100,"(A)") "-----------------------------------------------"
do i=1,n
    write(100,"(10000ES15.5)") a(i,:)
end do

err=1.
iter=0
!Write initial guess to output file
write(100,"(A)")  "---------------------------------"
write(100,"(A,)") "--------- Initial guess ----------"
write(100,"(A)")  "---------------------------------"
write(100,"(A,ES15.5)") "The error is: ",err

do i=1,n
    write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xold(i)
end do
```

```
!Computations for the Jacobi method
do while (iter<=maxiter .and. err>=rel)
    !Find new x
    do i=1,n
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xold(j)
        end do
        do j=i+1,n
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(a(i,n+1)-s)/a(i,i)
    end do
    !Find error
    err = sqrt(sum((xnew-xold)**2))/n
    xold = xnew
    iter=iter+1
    !Write results to output file
    write(100,"(A)") "---------------------------------"
    write(100,"(A,I5,A)") "--------- Iteration ",iter," ---------"
    write(100,"(A)") "---------------------------------"
    write(100,"(A,ES15.5)") "The error is : ",err
    do i=1,n
        write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xnew(i)
    end do
end do
!Write results to screen
If (iter>maxiter) then
    write(*,"(A)") "For x in each iteration open file Jacobi_results.dat..."
    write(*,*)
    write(*,"(A,I0,A)") "Solution didn't converge after: ",iter,"
iterations."
else
    write(*,"(A)") "For x in each iteration open file Jacobi_results.dat..."
    write(*,*)
    write(*,"(A,I0,A,ES12.4)") "Solution converged after: ",iter," iterations
with error: ",err
    write(*,"(A)") "Solution of linear system given below:"
    do i=1,n
        write(*,"(A2,I0,,A2,ES20.10)")"x(",i,")=",xnew(i)
    end do
endif
!Find program end time
call cpu_time(tend)
write(*,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."
write(100,"(A)") "---------------------------------------------------"
write(100,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."

!Close output file
close(100)
end
```

Τρέχοντας το πρόγραμμα φαίνεται πως η μέθοδος Jacobi με αρχική εκτίμηση $\boldsymbol{x} = 0$ συγκλίνει μετά από 38 επαναλήψεις με σφάλμα 9.25997×10^{-6}. Στον παρακάτω πίνακα φαίνεται πως διαμορφώνεται η λύση σε κάποιες ενδεικτικές επαναλήψεις.

	Αριθμός επαναλήψεων						
	1	2	5	10	20	30	38
x_1	189.14000	113.30969	182.64347	162.81042	164.67168	164.69937	164.69982
x_2	-24.20000	-24.20000	15.78054	-1.44981	-0.01681	0.00166	0.00196
x_3	43.93000	41.02440	24.32372	20.30527	20.68126	20.68536	20.68544
x_4	2.90560	2.90560	2.90560	2.90560	2.90560	2.90560	2.90560
x_5	0.00000	6.85300	20.94606	20.31435	20.33706	20.33890	20.33895
x_6	0.00000	51.39810	21.87734	24.70771	24.21222	24.20214	24.20198
x_7	95.79866	95.79866	95.79866	95.79866	95.79866	95.79866	95.79866
x_8	0.00000	2.78036	2.55457	2.43047	2.42159	2.42110	2.42109
x_9	14.61880	14.61880	14.61880	14.61880	14.61880	14.61880	14.61880
x_{10}	-99.10000	-42.12014	20.04860	-0.34589	0.02069	0.00021	-0.00028
x_{11}	10.56170	10.56170	10.56170	10.56170	10.56170	10.56170	10.56170
x_{12}	31.90031	27.37149	27.73927	27.94140	27.95588	27.95667	27.95669
x_{13}	8.40000	-97.41981	38.27063	6.81553	8.02518	8.04398	8.04457
x_{14}	97.90000	318.94031	299.45845	291.20042	290.59099	290.55725	290.55655

1β. Gauss-Seidel

Πρόγραμμα σε Fortran για την μέθοδο Gauss-Seidel:

```
program GaussSeidel
implicit none
integer::i,j,iter,maxiter,n
real*8:: s,rel,err,tstart,tend
real*8,allocatable:: a(:,:),xold(:),xnew(:)

!Find program start time
call cpu_time(tstart)

!Open output file
open (100,file="GaussSeidel_results.dat")

!Definition of parameters for the Gauss-Seidel method
n=14                                !Number of equations
allocate(a(n,n+1),xold(n),xnew(n))  !Allocation of required matrices
maxiter=10000                       !Maximum number of iteration
rel=1d-5                            !Desired relative error
xold=0.                             !Initial guess x0
```

```
!Definition of the linear system of equations
a(1,:)=(/1.,0.,1.,0.,0.,-1.,0.,0.,0.,0.,0.,1.,0.,0.,189.14/)
a(2,:)=(/0.,-1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,24.2/)
a(3,:)=(/0.,0.,1.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,43.93/)
a(4,:)=(/0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,2.9056/)
a(5,:)=(/0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,-0.07,0./)
a(6,:)=(/0.,0.,1.17,0.,0.,-1.,0.,0.,0.,0.,0.,0.,0.,0./)
a(7,:)=(/0.,0.,0.,0.,0.,0.,0.745,0.,0.,0.,0.,0.,0.,0.,71.37/)
a(8,:)=(/-0.0147,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0./)
a(9,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,14.6188/)
a(10,:)=(/0.,1.,0.,0.,1.,0.,1.,-1.,-1.,-1.,0.,0.,0.,0.,99.1/)
a(11,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,10.5617/)
a(12,:)=(/0.11,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,4.594,0.,0.,146.55/)
a(13,:)=(/0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.,-1.,-1.,0.,-8.4/)
a(14,:)=(/1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,-1.,-97.9/)

!Write system of equations to be solved to output file
write(100,"(A)") "-----------------------------------------------"
write(100,"(A)") "--------- Linear system of equations ----------"
write(100,"(A)") "-----------------------------------------------"
do i=1,n
    write(100,"(10000ES15.5)") a(i,:)
end do

err=1.
iter=0
xnew=xold

!Write initial guess to output file
write(100,"(A)")  "---------------------------------"
write(100,"(A,)") "--------- Initial guess ----------"
write(100,"(A)")  "---------------------------------"
write(100,"(A,ES15.5)") "The error is: ",err
do i=1,n
    write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xold(i)
end do

!Computations for the Gauss-Seidel method
do while (iter<=maxiter .and. err>=rel)

    !Find new x
    do i=1,n
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xnew(j)
        end do
        do j=i+1,n
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(a(i,n+1)-s)/a(i,i)
    end do
    !Find error
    err = sqrt(sum((xnew-xold)**2))/n
    xold = xnew
    iter=iter+1
```

```
    !Write results to output file
    write(100,"(A)") "---------------------------------"
    write(100,"(A,I5,A)") "--------- Iteration ",iter," ---------"
    write(100,"(A)") "---------------------------------"
    write(100,"(A,ES15.5)") "The error is : ",err

    do i=1,n
        write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xnew(i)
    end do

end do

!Write results to screen
If (iter>maxiter) then

    write(*,"(A)") "For x in each iteration open file
GaussSeidel_results.dat..."
    write(*,*)
    write(*,"(A,I0,A)") "Solution didn't converge after: ",iter,"
iterations."

Else

    write(*,"(A)") "For x in each iteration open file
GaussSeidel_results.dat..."
    write(*,*)
    write(*,"(A,I0,A,ES12.4)") "Solution converged after: ",iter," iterations
with error: ",err
    write(*,"(A)") "Solution of linear system given below:"

    do i=1,n
        write(*,"(A2,I0,,A2,ES20.10)")"x(",i,")=",xnew(i)
    end do

endif

!Find program end time
call cpu_time(tend)
write(*,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."
write(100,"(A)") "---------------------------------------------------"
write(100,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."

!Close output file
close(100)

end
```

Τρέχοντας το πρόγραμμα η μέθοδος Gauss-Seidel με αρχική εκτίμηση $\boldsymbol{x} = 0$ συγκλίνει μετά από μόνο 15 επαναλήψεις με σφάλμα 6.8801×10^{-7}. Στον πίνακα που ακολουθεί φαίνεται πως διαμορφώνεται η λύση σε κάποιες ενδεικτικές επαναλήψεις.

	Δείκτης επανάληψης						
	1	2	3	4	5	8	11
x_1	189.14000	169.23661	168.26609	164.50135	164.64239	164.70053	164.69983
x_2	-24.20000	27.19810	23.79855	-1.95175	-0.36070	0.00656	0.00191
x_3	43.93000	41.02440	19.01560	20.37547	20.44178	20.68837	20.68541
x_4	2.90560	2.90560	2.90560	2.90560	2.90560	2.90560	2.90560
x_5	0.00000	22.00880	20.64893	20.58262	20.32540	20.33904	20.33896
x_6	51.39810	47.99855	22.24825	23.83930	23.91689	24.20539	24.20193
x_7	95.79866	95.79866	95.79866	95.79866	95.79866	95.79866	95.79866
x_8	2.78036	2.48778	2.47351	2.41817	2.42024	2.42110	2.42109
x_9	14.61880	14.61880	14.61880	14.61880	14.61880	14.61880	14.61880
x_{10}	-44.90050	28.79899	24.05382	-1.70744	-0.37568	0.00437	-0.00036
x_{11}	10.56170	10.56170	10.56170	10.56170	10.56170	10.56170	10.56170
x_{12}	27.37149	27.84806	27.87130	27.96144	27.95806	27.95667	27.95669
x_{13}	-35.91113	37.01921	32.23654	6.32978	7.66700	8.04929	8.04454
x_{14}	314.41149	294.98467	294.03739	290.36280	290.50045	290.55721	290.55652

1γ. S.O.R.

Πρόγραμμα σε Fortran για την μέθοδο S.O.R.:

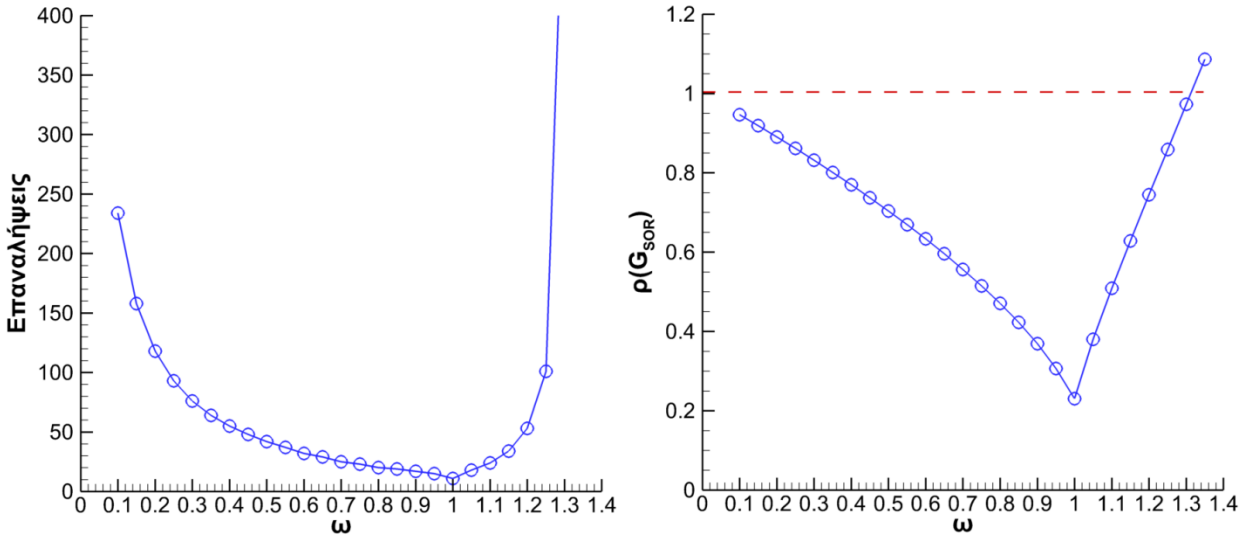
```
program SOR
implicit none
integer::i,j,iter,maxiter,n
real*8:: s,w,rel,err,tstart,tend
real*8,allocatable:: a(:,:),xold(:),xnew(:)

!Find program start time
call cpu_time(tstart)

!Open output file
open (100,file="SOR_results.dat")

!Definition of parameters for the SOR method
n=14                                !Number of equations
allocate(a(n,n+1),xold(n),xnew(n))  !Allocation of required matrices
maxiter=10000                       !Maximum number of iteration
rel=1d-5                            !Desired relative error
w=1.2                               !Relaxation parameter
xold=0.                             !Initial guess x0

!Definition of the linear system of equations
a(1,:)=(/1.,0.,1.,0.,0.,-1.,0.,0.,0.,0.,0.,1.,0.,0.,189.14/)
a(2,:)=(/0.,-1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,24.2/)
a(3,:)=(/0.,0.,1.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,43.93/)
a(4,:)=(/0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,2.9056/)
a(5,:)=(/0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,-0.07,0./)
```

```
a(6,:)=(/0.,0.,1.17,0.,0.,-1.,0.,0.,0.,0.,0.,0.,0.,0./)
a(7,:)=(/0.,0.,0.,0.,0.,0.,0.745,0.,0.,0.,0.,0.,0.,0.,71.37/)
a(8,:)=(/-0.0147,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0./)
a(9,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,14.6188/)
a(10,:)=(/0.,1.,0.,0.,1.,0.,1.,-1.,-1.,-1.,0.,0.,0.,0.,99.1/)
a(11,:)=(/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,10.5617/)
a(12,:)=(/0.11,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,4.594,0.,0.,146.55/)
a(13,:)=(/0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.,-1.,-1.,0.,-8.4/)
a(14,:)=(/1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,-1.,-97.9/)

!Write system of equations to be solved to output file
write(100,"(A)") "-------------------------------------------------"
write(100,"(A)") "--------- Linear system of equations ----------"
write(100,"(A)") "-------------------------------------------------"
do i=1,n
    write(100,"(10000ES15.5)") a(i,:)
end do

err=1.
iter=0
xnew=xold

!Write initial guess to output file
write(100,"(A)")  "---------------------------------"
write(100,"(A,)") "--------- Initial guess ----------"
write(100,"(A)")  "---------------------------------"
write(100,"(A,ES15.5)") "The error is: ",err
do i=1,n
    write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xold(i)
end do

!Computations for the SOR method
do while (iter<=maxiter .and. err>=rel)

    !Find new x
    do i=1,n
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xnew(j)
        end do
        do j=i+1,n
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(1.-w)*xold(i)+w*(a(i,n+1)-s)/a(i,i)
    end do

    !Find error
    err = sqrt(sum((xnew-xold)**2))/n
    xold = xnew
    iter=iter+1

    !Write results to output file
    write(100,"(A)") "---------------------------------"
    write(100,"(A,I5,A)") "--------- Iteration ",iter," ---------"
```

```
    write(100,"(A)") "---------------------------------"
    write(100,"(A,ES15.5)") "The error is : ",err

    do i=1,n
        write(100,"(2(A,I0),A,ES20.10)") "x",iter,"(",i,")=",xnew(i)
    end do
end do

!Write results to screen
If (iter>maxiter) then
    write(*,"(A)") "For x in each iteration open file SOR_results.dat..."
    write(*,*)
    write(*,"(A,ES12.4)") "Relaxation parameter: ",w
    write(*,"(A,I0,A)") "Solution didn't converge after: ",iter,"
iterations."
else
    write(*,"(A)") "For x in each iteration open file SOR_results.dat..."
    write(*,*)
    write(*,"(A,ES12.4)") "Relaxation parameter: ",w
    write(*,"(A,I0,A,ES12.4)") "Solution converged after: ",iter," iterations
with error: ",err
    write(*,"(A)") "Solution of linear system given below:"
    do i=1,n
        write(*,"(A2,I0,,A2,ES20.10)")"x(",i,")=",xnew(i)
    end do
endif

!Find program end time
call cpu_time(tend)
write(*,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."
write(100,"(A)") "-----------------------------------------------------"
write(100,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."

!Close output file
close(100)

end
```

Για την παραμετρική ανάλυση του αριθμού επαναλήψεων σε σχέση με την παράμετρο χαλάρωσης είναι αναγκαία η εκτέλεση του παραπάνω προγράμματος για διάφορες τιμές του ω. Στο γράφημα που ακολουθεί φαίνεται ότι ο ελάχιστος εφικτός αριθμός επαναλήψεων είναι 15 και αντιστοιχεί στην τιμή $\omega = 1$ (Gauss-Seidel) ενώ για $\omega > 1.3$ η μέθοδος S.O.R αρχίζει να αποκλίνει.

Για την κατανόηση αυτής της συμπεριφοράς είναι σημαντική η μελέτη της φασματικής ακτίνας του πίνακα επανάληψης της SOR. Κατασκευάζοντας το γράφημα της φασματικής ακτίνας σε σχέση με το ω γίνεται φανερό πως το βέλτιστο ω συμπίπτει με την ελάχιστη $\rho\left(\boldsymbol{G}_{SOR}\right)$ ενώ όταν $\rho\left(\boldsymbol{G}_{SOR}\right) > 1$ ή μέθοδος αποκλίνει.



Κώδικας σε Mathematica για τον υπολογισμό της φασματικής ακτίνας:

```
n = 14;
w = 0.1
A={{1.,0.,1.,0.,0.,-1.,0.,0.,0.,0.,0.,1.,0.,0.}, \
{0.,-1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.}, \
{0.,0.,1.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.}, \
{0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.}, \
{0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,-0.07}, \
{0.,0.,1.17,0.,0.,-1.,0.,0.,0.,0.,0.,0.,0.,0.}, \
{0.,0.,0.,0.,0.,0.,0.745,0.,0.,0.,0.,0.,0.,0.}, \
{-0.0147,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.}, \
{0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.}, \
{0.,1.,0.,0.,1.,0.,1.,-1.,-1.,-1.,0.,0.,0.,0.}, \
{0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.}, \
{0.11,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,4.594,0.,0.}, \
{0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.,-1.,-1.,0.}, \
{1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,-1.}};
Diag = ConstantArray[0., {n, n}];
Do[Diag[[i, i]] = A[[i, i]], {i, 1, n}];
Low = LowerTriangularize[A] - Diag;
Upp = UpperTriangularize[A] - Diag;
GSOR = IdentityMatrix[n] - w*Inverse[Diag + w*Low].A;
Max[Abs[Eigenvalues[GSOR]]]
```

Άσκηση 2

Το παρακάτω αλγεβρικό τριδιαγώνιο σύστημα προκύπτει από την διακριτοποίηση της συνήθους διαφορικής εξίσωσης μεταφοράς θερμότητας με αγωγή σε μια ράβδο με αδιάστατο μήκος $0 < x < 1$. Οι άγνωστοι του συστήματος T_i, $i = 1, 2, ..., N$ αντιστοιχούν στις θερμοκρασίες της ράβδου στα εσωτερικά ισαπέχοντα σημεία της ράβδου x_i, $i = 1, 2, ..., N$, ενώ οι ποσότητες A και B είναι οι θερμοκρασίες στα δύο άκρα της ράβδου στο $i = 0$ και $i = N + 1$ αντίστοιχα.

$$\begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & . & . & . & & & \\ & & 1 & -2 & 1 & & \\ & & & . & . & . & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ . \\ T_i \\ . \\ T_{N-1} \\ T_N \end{bmatrix} = \begin{bmatrix} -A \\ 0 \\ . \\ 0 \\ . \\ 0 \\ -B \end{bmatrix}, \quad i = 1, 2, ..., N$$

Να λυθεί το παραπάνω τριδιαγώνιο σύστημα για 3 εσωτερικά σημεία και θεωρώντας $A = 600$ και $B = 300$ με τις επαναληπτικές μεθόδους α) Jacobi β) Gauss-Seidel και γ) S.O.R. με $\omega = 1.2$. Τέλος, να γίνει παραμετρική μελέτη του απαιτούμενου αριθμού επαναλήψεων της S.O.R σε σχέση με το ω για $\mathrm{N} = 3,\ 30,\ 300$.

Σε όλες τις μεθόδους να χρησιμοποιηθεί κριτήριο τερματισμού: $\varepsilon = \frac{1}{N}\sqrt{\sum_{i=1}^{\mathrm{N}} \left(T_i^{(m+1)} - T_i^{(m)} \right)^2} < 10^{-5}$

Απάντηση:

Χρησιμοποιούνται οι κώδικες που δόθηκαν στην Άσκηση 1. Η μόνη αλλαγή είναι ο ορισμός του συστήματος:

```
!Definition of the linear system of equations
a=0. ; a(1,1)=-2 ; a(2,1)=1
Do i=2,n-1
    a(i,i)=-2
    a(i-1,i)=1
    a(i+1,i)=1
Enddo
a(n,n)=-2; a(n-1,n)=1; a(1,n+1)=-600; a(n,n+1)=-300
```

Για τις φασματικές ακτίνες των πινάκων επανάληψης μπορεί να χρησιμοποιηθεί ο παρακάτω κώδικας σε Mathematica. Αρχικά ο πίνακας ***A*** σπάει στους πίνακες ***D*** (διαγώνιος), ***L*** (κάτω τριγωνικός) και ***U*** (άνω τριγωνικός). Κατασκευάζεται ο πίνακας επανάληψης ***G*** κάθε μεθόδου, υπολογίζονται όλες οι ιδιοτιμές του και εμφανίζεται η ιδιοτιμή με το μεγαλύτερο μέτρο.

```
n = 3;
w = 1.2;
A = {{-2, 1, 0}, {1, -2, 1}, {0, 1, -2}};
Diag = ConstantArray[0., {n, n}];
Do[Diag[[i, i]] = A[[i, i]], {i, 1, n}]
Low = LowerTriangularize[A] - Diag;
Upp = UpperTriangularize[A] - Diag;
GJ = IdentityMatrix[n] - Inverse[Diag].A;
GGS = IdentityMatrix[n] - Inverse[Diag + Low].A;
GSOR = IdentityMatrix[n] - w*Inverse[Diag + w*Low].A;
MatrixForm[GJ]
MatrixForm[GGS]
MatrixForm[GSOR]
```

```
Det[GJ - IdentityMatrix[n]*\[Lambda]]
Det[GGS - IdentityMatrix[n]*\[Lambda]]
Det[GSOR - IdentityMatrix[n]*\[Lambda]]
Max[Abs[Eigenvalues[GJ]]]
Max[Abs[Eigenvalues[GGS]]]
Max[Abs[Eigenvalues[GSOR]]]
```

Με βάση τα παραπάνω προκύπτουν οι πίνακες επανάληψης των τριών μεθόδων:

$$\boldsymbol{G}_J = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix}, \quad \boldsymbol{G}_{GS} = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & 0.25 & 0.5 \\ 0 & 0.125 & 0.25 \end{bmatrix}, \quad \boldsymbol{G}_{SOR} = \begin{bmatrix} -0.2 & 0.6 & 0 \\ -0.12 & 0.16 & 0.6 \\ -0.072 & 0.096 & 0.16 \end{bmatrix}$$

Οι αντίστοιχες χαρακτηριστικές εξισώσεις είναι:

$$-\lambda_J^3 + 0.5\lambda_J = 0, \quad -\lambda_{GS}^3 + 0.5\lambda_{GS}^2 = 0, \quad -\lambda_{SOR}^3 + 0.12\lambda_{SOR}^2 + 0.024\lambda_{SOR} - 0.008 = 0$$

Από την λύση των χαρακτηριστικών εξισώσεων προκύπτουν $\rho(\boldsymbol{G}_J) = 0.7071$, $\rho(\boldsymbol{G}_{GS}) = 0.5$ και $\rho(\boldsymbol{G}_{SOR}) = 0.2$. Συμπεραίνεται πως και οι τρεις μέθοδοι θα συγκλίνουν.

2α. Jacobi

Για την μέθοδο Jacobi με αρχική εκτίμηση $\boldsymbol{T} = 0$ συγκλίνει μετά από 48 επαναλήψεις με σφάλμα 8.9407×10^{-6}. Παρακάτω δίνεται η λύση σε κάποιες ενδεικτικές επαναλήψεις.

	Δείκτης επανάληψης					
	1	2	5	10	20	48
$T(0.25)$	300.0	300.0	468.75	510.9375	524.56055	524.99997
$T(0.50)$	0.000	225.0	337.5	435.9375	449.56055	449.99997
$T(0.75)$	150.0	150.0	318.75	360.9375	374.56055	374.99997

2β. Gauss-Seidel

Για την μέθοδο Gauss-Seidel με αρχική εκτίμηση $\boldsymbol{x} = 0$ συγκλίνει μετά από 25 επαναλήψεις με σφάλμα 8.9407×10^{-6}. Παρακάτω παρουσιάζεται η λύση σε κάποιες ενδεικτικές επαναλήψεις.

	Δείκτης επανάληψης					
	1	2	5	10	15	25
$T(0.25)$	300.0	375.0	506.250	524.4141	524.98169	524.99998
$T(0.50)$	150.0	300.0	431.250	449.4141	449.98169	449.99998
$T(0.75)$	225.0	300.0	365.625	374.7070	374.99085	374.99999

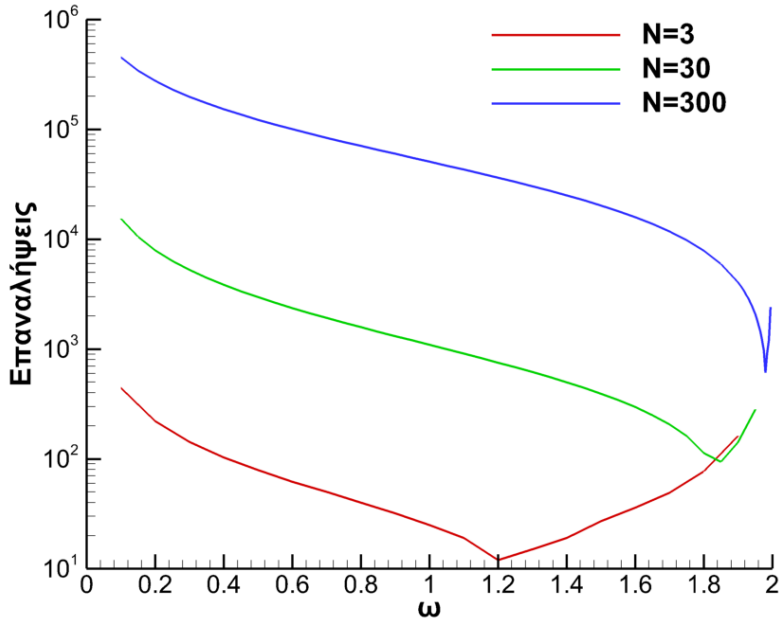
2γ. S.O.R

Για την μέθοδο S.O.R με παράμετρο χαλάρωσης $\omega = 1.2$ και αρχική εκτίμηση $\boldsymbol{T} = 0$ συγκλίνει μετά από μόνο 12 επαναλήψεις με σφάλμα 2.8997×10^{-6}. Στον πίνακα που ακολουθεί δίνεται η λύση σε κάποιες ενδεικτικές επαναλήψεις.

	Δείκτης επανάληψης					
	1	2	3	5	8	12
$T(0.25)$	360.0	417.60	512.3520	525.22261	525.00388	524.999989
$T(0.50)$	216.0	393.12	441.1584	450.17635	450.00265	449.999994
$T(0.75)$	309.6	353.95	373.9046	375.12849	375.00028	374.999998

2δ. Παραμετρική ανάλυση αριθμού επαναλήψεων S.O.R

Τρέχοντας το κώδικα της SOR για διάφορες τιμές του ω και για $N = 3,\ 30,\ 300$ μπορεί να κατασκευαστεί το γράφημα του αριθμού επαναλήψεων σε σχέση με το ω. Φαίνεται ότι για $N = 3$ το βέλτιστο ω είναι ~1.2, ενώ καθώς το μέγεθος του συστήματος αυξάνει το βέλτιστο ω επίσης αυξάνει. Για $N = 30$ και $N = 300$ τα βέλτιστα ω είναι ~1.85 και ~1.98 αντίστοιχα.



Άσκηση 3

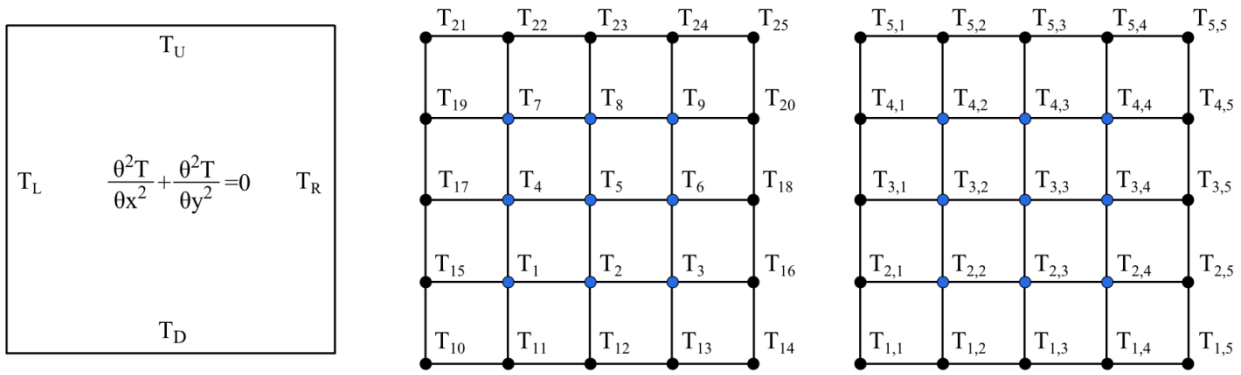
Η μετάδοση μεταφοράς θερμότητας σε ένα τετράγωνο $0 < x, y < 1$ στο οποίο δεν υπάρχει παραγωγή θερμότητας περιγράφεται από την παρακάτω ελλειπτική ΜΔΕ:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Αν με $T_{i,j}$ συμβολιστεί η θερμοκρασία στον κόμβο i, j τότε η αριθμητική επίλυση της καταλήγει στην επίλυση ενός γραμμικού συστήματος της μορφής:

$$4T_{i,j} - T_{i+1,j} - T_{i-1,j} - T_{i,j+1} - T_{i,j+1} = 0$$

Στην παρακάτω εικόνα αναπαριστάται γραφικά το παραπάνω πρόβλημα για $N \times N = 3 \times 3 = 9$ όπου φαίνονται οι 9 άγνωστες θερμοκρασίες (μπλε). Οι θερμοκρασίες στο σύνορο του τετραγώνου είναι γνωστές και συμβολίζονται με T_D, T_U, T_R και T_L για την κάτω, πάνω, αριστερή και δεξιά πλευρά του τετραγώνου αντίστοιχα.



Σύμφωνα με την αρίθμηση που φαίνεται παραπάνω το γραμμικό σύστημα που πρέπει να λυθεί:

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{pmatrix} = \begin{pmatrix} T_{11} + T_{15} \\ T_{12} \\ T_{13} + T_{16} \\ T_7 \\ 0 \\ T_{18} \\ T_{19} + T_{22} \\ T_{23} \\ T_{20} + T_{24} \end{pmatrix} = \begin{pmatrix} T_D + T_L \\ T_D \\ T_D + T_R \\ T_L \\ 0 \\ T_R \\ T_L + T_U \\ T_U \\ T_R + T_U \end{pmatrix}$$

Να λυθεί το γραμμικό σύστημα που προκύπτει από την εξίσωση θερμότητας σε ένα τετράγωνο με $T_D = T_L = T_R = 0$ και $T_U = 100$ για $N = 3,\ 9,\ 19$ με τις μεθόδους Jacobi, Gauss-Seidel και S.O.R και να δοθεί σαν αποτέλεσμα ο αριθμός των απαιτούμενων επαναλήψεων. Να γίνει το γράφημα της κατανομής θερμοκρασίας κατά μήκος του $0 \le y \le 1$ για $x = 0.5$. Τέλος να κατασκευαστεί η γραφική παράσταση των ισοϋψών θερμοκρασίας για $N = 99$.

Κριτήριο τερματισμού $\varepsilon = \dfrac{1}{N \times N} \sqrt{\sum_{i=1}^{N \times N} \left(T_i^{(m+1)} - T_i^{(m)} \right)^2} < 10^{-6}$

Απάντηση:

Η παραπάνω ΜΔΕ μπορεί να επιλυθεί αναλυτικά σε Mathematica:

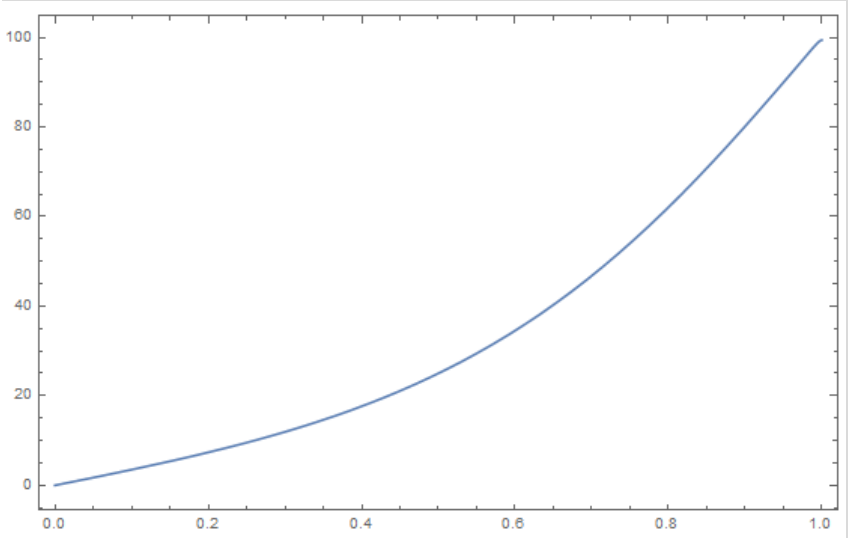
```
sol = DSolve[{D[T[x, y], {x, 2}] + D[T[x, y], {y, 2}] == 0, \
T[0, y] == 0, T[x, 0] == 0, T[1, y] == 0, T[x, 1] == 100}, T[x, y],x, y]
```

Αναλυτική λύση:

$$T(x,y) = \sum_{n=1}^{\infty} -\frac{200\left(-1+(-1)^k\right)\sin(n\pi x)\sinh(n\pi y)}{n\pi\cosh(n\pi)}$$

Δίδεται η αναλυτική γραφική παράσταση της θερμοκρασίας για $x = 0.5$ και $0 \le y \le 1$ διατηρώντας τους πρώτους 100 όρους της απειροσειράς:

```
Plot[Evaluate[T[x, y] /. sol[[1]]] /. {x -> 0.5, Infinity -> 100} \
// Activate, {y, 0, 1}, Frame -> True]
```



Κώδικας σε Fortran που επιλύει το παραπάνω πρόβλημα για οποιαδήποτε N με τις μεθόδους Jacobi, Gauss-Seidel και S.O.R. :

```
program HeatTransferSquare
implicit none
integer::i,j,k,n,nn,maxiter,solver
integer,allocatable::numij(:,:)
real*8:: w,rel,tstart,tend,TD,TL,TR,TU
real*8,allocatable:: a(:,:),x0(:),x(:),tempDistr(:,:)

!Find program start time
call cpu_time(tstart)

!Open output file
open (100,file="HeatTransferSquare_results.dat")

!Definition of parameters
solver=3          !1:Jacobi, 2:Gauss-Seidel, 3:SOR
n=99              !Number of inside nodes per square side
nn=n*n                  !Number of equations
TD=0.             !temperature of plate down side
TL=0.             !temperature of plate left side
TR=0.             !temperature of plate right side
TU=100.                 !temperature of plate up side
allocate(a(nn,nn+1),x0(nn),x(nn),numij(nn,2),tempDistr(n+2,n+2))
maxiter=1000000   !Maximum number of iteration
rel=1d-6          !Desired relative error
w=1.9             !Relaxation parater (used only in SOR)
x0=0.             !Initial guess x0

!Definition of linear system of equations
do i=1,n
    do j=1,n
        k=(i-1)*n+j
        numij(k,1)=i
        numij(k,2)=j
    end do
end do

a=0.
do i=1,nn
    if (numij(i,1)==1 .and. numij(i,2)==1) then
    !Down-left corner
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i+n)=-1.; a(i,nn+1)=TD+TL
    else if (numij(i,1)==1 .and. numij(i,2)==n) then
    !Down-right corner
    a(i,i)=4.; a(i,i-1)=-1.; a(i,i+n)=-1.; a(i,nn+1)=TD+TR
    else if (numij(i,1)==n .and. numij(i,2)==1) then
    !Up-left corner
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i-n)=-1.; a(i,nn+1)=TU+TL
    else if (numij(i,1)==n .and. numij(i,2)==n) then
    !Up-right corner
    a(i,i)=4.; a(i,i-1)=-1.; a(i,i-n)=-1.; a(i,nn+1)=TU+TR
    else if (numij(i,1)==1) then
    !Down side
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i-1)=-1.; a(i,i+n)=-1.; a(i,nn+1)=TD
    else if (numij(i,1)==n) then
```

```
    !Up side
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i-1)=-1.; a(i,i-n)=-1. ; a(i,nn+1)=TU
    elseif (numij(i,2)==1) then
    !left side
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i+n)=-1.; a(i,i-n)=-1.; a(i,nn+1)=TL
    else if (numij(i,2)==n) then
    !Right side
    a(i,i)=4.; a(i,i-1)=-1.; a(i,i+n)=-1.; a(i,i-n)=-1.; a(i,nn+1)=TR
    else
    !Inside
    a(i,i)=4.; a(i,i+1)=-1.; a(i,i-1)=-1.; a(i,i+n)=-1.; a(i,i-n)=-1.
    end if
end do
write(*,"(A)") "Solving..."
!Call the appropriate solver to solve the system of equations
If (Solver==1) then
   Call Jacobi(nn,a,x,x0,rel,maxiter)
else if (Solver==2) then
   Call GaussSeidel(nn,a,x,x0,rel,maxiter)
else if (Solver==3) then
   Call SOR(nn,a,x,x0,w,rel,maxiter)
end if

!Arrange and write results to screen and file
tempDistr(1,:)=TD
do i=2,n+1
      tempDistr(i,1)=TL
    tempDistr(i,2:(n+1))=x((i-2)*n+1:(i-1)*n)
    tempDistr(i,n+2)=TR
end do
tempDistr(n+2,:)=TU

if (mod(n,2)==1) then
    write(100,"(A)") "Temperature profile for x=0.5"
    do i=1,n+2
    write(100,"(2ES15.5)") (i-1)/Real(n+1),TempDistr(i,(n+3)/2)
    end do
else
    write(*,*) "n must be odd to find temperature at (x=0.5,y=0.5)"
end if
write(100,*)
write(100,"(A)") "Temperature distribution for whole square"
do i=1,n+2
    write(100,"(1000F9.4)") tempDistr(i,:)
enddo

!Find program end time
call cpu_time(tend)
write(*,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."
write(100,"(A)") "-------------------------------------------------"
write(100,"(A,ES14.4,2X,A)") "Program has used", tend-tstart,"seconds of CPU
time."
write(*,"(A)") "Open file HeatTransferSquare_results.dat to find"
write(*,"(A)") "detailed temperature distribution..."
```

```
Contains

Subroutine Jacobi(nn,a,xnew,x0,rel,maxiter)
integer::i,j,iter,nn,maxiter
real*8:: s,err,a(nn,nn+1),xnew(nn),x0(nn),rel
real*8,allocatable::xold(:)

allocate(xold(nn))
xold=x0
err=1.
iter=0

do while (iter<=maxiter .and. err>=rel)
    !Find new x
    do i=1,nn
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xold(j)
        end do
        do j=i+1,nn
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(a(i,nn+1)-s)/a(i,i)
    end do
    !Find error
    err = sqrt(sum((xnew-xold)**2))/nn
    xold = xnew
    iter=iter+1
    if (mod(iter,10)==0) write(*,"(2(A,1X,I0,1X,A,ES15.5))") "Jacobi
iter:",iter,"error:",err
end do
write(*,*)
if (iter>=maxiter) then
    write(*,"(A,1X,I0,1X,A)") "Jacobi didn't converge
after",maxiter,"iterations."
else
    write(*,"(A,1X,I0,1X,A)") "Jacobi converged after",iter,"iterations"
    write (*,"(A,ES15.5)")"with error",err
endif
deallocate(xold)
end Subroutine Jacobi

Subroutine GaussSeidel(nn,a,xnew,x0,rel,maxiter)
integer::i,j,iter,nn,maxiter
real*8:: s,err,a(nn,nn+1),xnew(nn),x0(nn),rel
real*8,allocatable::xold(:)

allocate(xold(nn))
xold=x0
err=1.
iter=0
do while (iter<=maxiter .and. err>=rel)
   !Find new x
```

```
    do i=1,nn
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xnew(j)
        end do
        do j=i+1,nn
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(a(i,nn+1)-s)/a(i,i)
    end do
    !Find error
    err = sqrt(sum((xnew-xold)**2))/nn
    xold = xnew
    iter=iter+1
    if (mod(iter,10)==0) write(*,"(2(A,1X,I0,1X,A,ES15.5))") "Gauss-Seidel
iter:",iter,"error:",err
end do
deallocate(xold)
write(*,*)
if (iter>=maxiter) then
    write(*,"(A,1X,I0,1X,A)") "Gauss-Seidel didn't converge
after",maxiter,"iterations."
else
    write(*,"(A,1X,I0,1X,A)") "Gauss-Seidel converged
after",iter,"iterations"
    write (*,"(A,ES15.5)")"with error",err
endif
end Subroutine GaussSeidel

Subroutine SOR(nn,a,xnew,x0,w,rel,maxiter)
integer::i,j,iter,nn,maxiter
real*8:: s,w,err,a(nn,nn+1),xnew(nn),x0(nn),rel
real*8,allocatable::xold(:)

allocate(xold(nn))
xold=x0
err=1.
iter=0
do while (iter<=maxiter .and. err>=rel)
    !Find new x
    !Find new x
    do i=1,nn
        s=0.
        do j=1,i-1
            s=s+a(i,j)*xnew(j)
        end do
        do j=i+1,nn
            s=s+a(i,j)*xold(j)
        end do
        xnew(i)=(1-w)*xold(i)+w*(a(i,nn+1)-s)/a(i,i)
    end do
    !Find error
    err = sqrt(sum((xnew-xold)**2))/nn
    xold = xnew
    iter=iter+1
```

```
    if (mod(iter,10)==0) write(*,"(2(A,1X,I0,1X,A,ES15.5))") "SOR
iter:",iter,"error:",err
end do
deallocate(xold)
write(*,*)
if (iter>=maxiter) then
    write(*,"(A,1X,I0,1X,A)") "SOR didn't converge
after",maxiter,"iterations."
else
    write(*,"(A,1X,I0,1X,A)") "SOR converged after",iter,"iterations"
    write (*,"(A,ES15.5)")"with error",err
endif
end Subroutine SOR

end
```

Δίνεται ο απαιτούμενος αριθμός επαναλήψεων για τις μεθόδους Jacobi, Gauss-Seidel και SOR για διάφορα ω και για $N = 3,\ 9,\ 19$.

N	Jacobi	Gauss-Seidel	SOR				
			$\omega = 1.1$	$\omega = 1.3$	$\omega = 1.5$	$\omega = 1.7$	$\omega = 1.9$
3	44	24	19	16	26	47	157
9	235	127	105	69	38	46	145
19	771	419	350	239	152	74	140

Τα αποτελέσματα των τριών μεθόδων στα αντίστοιχα σημεία. Εφόσον για $N = 19$ το βέλτιστο ω είναι κοντά στο 1.7 είναι λογικό για την επίλυση του συστήματος $N \times N = 99 \times 99 = 9801$ να επιλεγεί η μέθοδος SOR με $\omega = 1.9$ για να επιτευχθεί ο ελάχιστος αριθμός επαναλήψεων. Οι δύο γραφικές παραστάσεις της αριθμητικής λύσης $T(0.5, y)$ για $N = 3,\ 9,\ 19$ καθώς και των ισοϋψών θερμοκρασίας για $N = 99$ δίνονται στα παρακάτω γραφήματα.