Scilab Textbook Companion for DC Machines and Synchronous Machines by U. A. Bakshi and M. V. Bakshi[1]

Created by
S. Sai Ashrith Reddy
B.Tech 3rd Year
Electrical Engineering
NIT, Karnataka
College Teacher
NA
Cross-Checked by
Chaitanya

May 25, 2016

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: DC Machines and Synchronous Machines

Author: U. A. Bakshi and M. V. Bakshi

Publisher: Technical Publications, Pune

Edition: 1

Year: 2008

ISBN: 9788184314830

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes 4

1 DC Generators 6

2 DC Motors 48

3 Testing of DC Macines 96

4 Synchronous Machines Alternators 141

5 Methods for Calculating Regulation of Alternator 162

6 Synchronization and Parallel Operation of Alternators 201

7 Synchronous Motors 254

List of Scilab Codes

Exa 1.1 TO DETERMINE EMF GENERATED DUE TO ROTATION AND REPLACEMENT OF LAP WOUND ARMATURE WITH WAVE WOUND 6
Exa 1.2 TO DETERMINE GENERATED EMF AND THE SPEED TO GENERATE THE SAME EMF USING WAVE WOUND ARMATURE 7
Exa 1.3 TO DRAW A DEVELOPED DIAGRAM FOR GENERATOR 7
Exa 1.4 TO DRAW DEVELOPED DIAGRAM FOR A DC GENERATOR 10
Exa 1.5 TO CALCULATE DEMAGNETISING AND CROSSMAGNETISING AMPERE TURNS PER POLE . . . 13
Exa 1.6 TO DETERMINE NUMBER OF COMPENSATING CONDUCTORS PER POLE 13
Exa 1.7 TO FIND REACTIVE VOLTAGE DURING LINEAR AND SINUSOIDAL COMMUTATION 14
Exa 1.8 TO FIND INDUCED EMF IN A GENERATOR . . . 15
Exa 1.9 TO DETERMINE ARMATURE RESISTANCE OF GENERATOR 16
Exa 1.10 TO DETERMINE TERMINAL VOLTAGE AT THE LOAD 17
Exa 1.11 TO CALCULATE THE VOLTAGE GENERATED BY SHUNT COMPOUND DC GENERATOR 19
Exa 1.12 TO CALCULATE THE OPEN CIRCUIT VOLTAGE AND LOAD CURRENT 20
Exa 1.13 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR USING ITS MAGNETISING CURVE 22

Exa 1.14 TO DETERMINE RUNNING SPEED TO GENERATE 240 V ON NOLOAD 24
Exa 1.15 TO CALCULATE LOAD CURRENT 24
Exa 1.16 TO CALCULATE ARMATURE CURRENT AND GENERATED EMF 26
Exa 1.17 TO DETERMINE THE TERMINAL VOLTAGE . . . 26
Exa 1.18 TO DETERMINE THE DRIVING SPEED OF ARMATURE TO GENERATE CERTAIN EMF 28
Exa 1.19 TO CALCULATE CERTAIN QUANTITIES FROM OPEN CIRCUIT CHARACTERISTICS OF DC SHUNT MOTOR 30
Exa 1.20 TO CALCULATE AMPERE TURNS AND SERIES TURNS TO BALANCE DEMAGNETISING COMPONENT OF A LAP CONNECTED GENERATOR . . 30
Exa 1.21 TO DESIGN A LAP WINDING 31
Exa 1.22 TO DETERMINE CERTAIN QUANTITIES ASSOCIATED WITH SIMPLEX WAVE WOUND DC MACHINE 34
Exa 1.23 TO DRAW DEVELOPED ARMATURE WINDING DIAGRAM OF DC MACHINE 35
Exa 1.24 TO DETERMINE REACTIVE VOLTAGE IN CASE OF LINEAR AND SINUSOIDAL COMMUTATION . 38
Exa 1.25 TO CALCULATE ARMATURE CURRENT AND OUTPUT POWER 39
Exa 1.26 TO DETERMINE REACTIVE VOLTAGE FOR A DC MACHINE 41
Exa 1.27 TO CALCULATE CROSS AND DEMAGNETISING TURNS PER POLE 42
Exa 1.28 TO CALCULATE REACTIVE VOLTAGE IN CASE OF LINEAR COMMUTATION 43
Exa 1.29 TO CALCULATE DEMAGNETISING AND CROSS MAGNETISING AMPERE TURNS PER POLE . . . 43
Exa 1.30 TO CALCULATE ARMATURE REACTION AMPERE TURNS AND DEMAGNETISING AND CROSSMAGENTISING AMPERE TURNS 45
Exa 1.31 TO DETERMINE ALTERED CURRENT WHEN SPEED OF SEPERATELY EXCITED GENERATOR IS DROPPED 46
Exa 2.1 TO DETERMINE INDUCED EMF IN MOTOR . . . 48

Exa 2.2 TO CALCULATE BACK EMF AND MOTOR SPEED 48
Exa 2.3 TO DETERMINE GROSS TORQUE DEVELOPED BY MOTOR ARMATUTRE 49
Exa 2.4 TO DETERMINE CERTAIN QUANTITIES RELATED TO LAP WOUND DC MOTOR 49
Exa 2.5 TO CALCULATE SPEED WHEN MOTOR DRAWS 60 A FROM SUPPLY 50
Exa 2.6 TO DETERMINE ARMATURE CURRENT AND BACK EMF 53
Exa 2.7 TO DETERMINE SPEED ON FULL LOAD 53
Exa 2.8 TO DETERMINE SPEED OF MOTOR WITH ALTERED LOAD 55
Exa 2.9 TO DETERMINE ARMATURE CURRENT WHEN A RESISTANCE IS ADDED IN SERIES TO FIELD WINDING 57
Exa 2.10 TO DETERMINE MOTOR SPEED WHEN FIELD WINDING GETS SHUNTED BY A RESISTANCE 58
Exa 2.11 TO DETERMINE EXTRA RESISTANCE THAT WILL REDUCE THE SPEED 59
Exa 2.12 TO DETERMINE CERTAIN QUANTITIES RELATED TO PERMANENT MAGNET DC MOTOR 61
Exa 2.13 TO DETERMINE SPEED ON HALF LOAD CONDITION 62
Exa 2.14 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR 62
Exa 2.15 TO DETERMINE MECHANICAL POWER AND NOLOAD SPEED AND CURRENT 64
Exa 2.16 TO DETERMINE FULL LOAD SPEED 65
Exa 2.17 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR 66
Exa 2.18 TO DETERMINE SPEED IF FIELD WINDING IS SHUNTED BY ADDITIONAL RESISTANCE 67
Exa 2.19 TO DETERMINE SPEED IF FIELD GROUPS ARE ARRANGED IN PARALLEL 69
Exa 2.20 TO DETERMINE NEW SPEED AND ARMATURE CURRENT AFTER RECONNECTION 71

Exa 2.21 TO PROVE THAT PROPORTIONALITY CONSTANT IS SAME IN CASE OF BACK EMF and ARMATURE SPEED AND TORQUE AND ARMATURE CURRENT 73
Exa 2.22 TO CALCULATE EXTRA RESISTANCE TO REDUCE THE SPEED 74
Exa 2.23 TO DETERMINE ADDITIONAL RESISTANCE IN FIELD CIRCUIT TO RAISE THE SPEED 74
Exa 2.24 TO DETERMINE SUPPLY VOLTAGE REQUIRED TO RAISE FAN SPEED 77
Exa 2.25 TO CALCULATE RESISTANCE TO BE CONNECTED IN SERIES WITH ARMATURE TO HALVE THE SPEED 78
Exa 2.26 TO CALCULATE TORQUE ALTERED DUE TO CHANGES IN FIELD FLUX AND ARMATURE CURRENT . . 79
Exa 2.27 TO CALCULATE EXTRA RESISTANCE IN SERIES WITH ARMATURE TO REDUCE SPEED AT FULL LOAD 79
Exa 2.28 TO DETERMINE SPEED WHEN DC SHUNT MOTOR GETS LOADED 81
Exa 2.29 TO DETERMINE SPEED AND TORQUE DEVELOPED AT FULL LOAD WHEN NO LOAD FLUX WEAKENS 81
Exa 2.30 TO FIND THE SPEED WHEN ADDITIONAL RESISTANCES GET CONNECTED WITH SHUNT FIELD AND ARMATURE 82
Exa 2.31 TO DETERMINE EXTRA RESISTANCE WITH FIELD CURRENT TO INCREASE SPEED OF DC SHUNT MOTOR 84
Exa 2.32 TO FIND EXTRA RESISTANCE TO BE ADDED IN SERIES WITH ARMATURE TO REDUCE ITS SPEED WITH SAME ARMATURE CURRENT 86
Exa 2.33 TO DETERMINE THE SPEED WHEN ADDITIONAL RESISTANCE GETS CONNECTED AND DRAWING SAME CURRENT 86
Exa 2.34 TO DETERMINE ADDITIONAL RESISTANCE IN SERIES WITH ARMATURE TO REDUCE THE SPEED AND ALTERED SPEED WHEN TORQUE GETS HALVED 88

Exa 2.35 TO CALCULATE SPEED AND USEFUL TORQUE ON FULL LOAD 89
Exa 2.36 TO DETERMINE MOTOR SPEED IF ADDITIONAL RESISTANCE IS INSERTED IN SERIES WITH ARMATURE CIRCUIT 90
Exa 2.37 TO DETERMINE RESISTANCE TO BE INSERTED IN SHUNT FIELD CIRCUIT TO INCREASE THE SPEED 91
Exa 2.38 TO DETERMINE TORQUES BEFORE AND AFTER FIELD WEAKENING 92
Exa 2.39 TO DETERMINE STALLING TORQUE AND TORQUES ON FULL LOAD AND DOUBLE FULL LOAD . . . 93
Exa 2.40 TO DETERMINE SPEED OF MOTOR FULL LOAD TORQUE AND MULTIPLES OF FULL LOAD TORQUE 94
Exa 3.1 TO DETERMINE CERTAIN QUANTITIES RELATED TO WAVE CONNECTED SHUNT MOTOR 96
Exa 3.2 TO DETERMINE FULL LOAD FULL LOAD OUTPUT AND EFFICIENCY 97
Exa 3.3 TO ESTIMATE FULL LOAD CURRENT AND EFFICIENCY 98
Exa 3.4 TO CALCULATE FULL LOAD EFFICIENCY OF DC SHUNT MOTOR 99
Exa 3.5 TO FIND THE EFFICIENCY OF MOTOR 99
Exa 3.6 TO DETERMINE THE EFFICIENCY OF MACHINES 100
Exa 3.7 TO FIND EFFICIENCY OF EACH MACHINE . . . 102
Exa 3.8 TO DETERMINE EFFICIENCY WHEN MOTOR DRAWS 100 A CURRENT 105
Exa 3.9 TO DETERMINE EFFICIENCY AND PERCENTAGE CHANGE IN SPEED 106
Exa 3.10 TO DETERMINE EFFICIENCY AND SPEED WHEN MOTOR DRAWS CERTAIN CURRENT 107
Exa 3.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 250 V DC HUNT MOTOR 108
Exa 3.12 TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V SHUNT MOTOR 110
Exa 3.13 TO DETERMINE CERTAIN QUANTITIES RELATED TO 240 V DC SHUNT MOTOR 111

Exa 3.14 TO DETERMINE FULL LOAD OUTPUT AND EFFICIENCY 112
Exa 3.15 TO CALCULATE MACHINE EFFICIENCY WHEN OPERATING AS A GENERATOR 113
Exa 3.16 TO DETERMINE FULL LOAD OUTPUT POWER EFFICIENCY AND PERCENTAGE CHANGE IN SPEED 114
Exa 3.17 TO DETERMINE EFFICIENCY AND PERCENTAGE CHANGE IN SPEED OF A SHUNT MOTOR 115
Exa 3.18 TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V DC SHUNT MOTOR 116
Exa 3.19 TO DETERMINE CERTAIN QUANTITIES AFTER PERFORMING RETARDATION TEST ON DC MACHINE 118
Exa 3.20 TO DETERMINE CERTAIN QUANTITIES AFTER PERFORMING RETARDATION TEST ON DC MACHINE 119
Exa 3.21 TO DETERMINE EFFICIENCY WHEN MACHINE IS OPERATED AS MOTOR 120
Exa 3.22 TO DETERMINE STRAY LOSSES OF MOTOR . . 121
Exa 3.23 TO DETERMINE EFFICIENCY OF EACH OF THE 2 SHUNT MACHINES 121
Exa 3.24 TO DETERMINE EFFICIENCY OF MOTOR AND GENERATOR 123
Exa 3.25 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES 125
Exa 3.26 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR ON FULL LOAD 127
Exa 3.27 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES 129
Exa 3.28 TO CALCULATE EFFICIENCY OF MACHINE ACTING AS GENERATOR 131
Exa 3.29 TO CALCULATE EFFICIENCY OF DC MACHINES 133
Exa 3.30 TO ESTIMATE EFFICIENCY OF 2 DC MACHINES 135
Exa 3.31 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR 136
Exa 3.32 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR 138

Exa 4.1 TO DRAW THE DIAGRAM FOR FULL PITCH ARMATURE WINDING OF AN ALTERNATOR 141
Exa 4.2 TO CALCULATE DISTRIBUTION FACTOR OF THREE PHASE ALTERNATOR 141
Exa 4.3 TO CALCULATE COIL SPAN FACTOR OF ARMATURE WINDING 142
Exa 4.4 TO CALCULATE INDUCED EMF ACROSS THE TERMINALS 143
Exa 4.5 TO DETERMINE FREQUENCY OF INDUCED EMF and FLUX PER POLE 144
Exa 4.6 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE ALTERNATOR 145
Exa 4.7 TO CALCULATE THE FLUX PER POLE OF 3 PHASE STAR CONNECTED ALTERNATOR 147
Exa 4.8 TO CALCULATE THE INDUCED EMF OF 1 PHASE ALTERNATOR 148
Exa 4.9 TO DETERMINE INDUCED EMF BETWEEN THE LINES OF 3 PHASE STAR CONNECTED ALTERNATORS 149
Exa 4.10 TO DETERMINE CERTAIN QUANTITIES RELATED TO 12 POLE 3 PHASE STAR CONNECTED ALTERNATOR 149
Exa 4.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED ALTERNATORS 151
Exa 4.12 TO DETERMINE INDUCED EMF IN 3 PHASE ALTERNATOR 152
Exa 4.13 TO CALCULATE FREQUENCY AND LINE VOLTAGE OF 3PHASE ALTERNATOR 153
Exa 4.14 TO DETERMINE kVA RATING OF A SYNCHRONOUS GENERATOR 154
Exa 4.15 TO DETERMINE THE NUMBER OF ARMATURE CONDUCTORS REQUIRED TO GIVE A LINE VOLTAGE OF 11kV 155
Exa 4.16 TO DETERMINE RMS VALUE OF PHASE AND LINE VOLTAGE 156
Exa 4.17 TO DETERMINE RESULTANT PHASE VOLTAGE AND LINE VOLTAGE 158

Exa 4.18 TO DETERMINE THE RATINGS WHEN DELTA CONNECTED ALTERNATOR IS RECONNECTED IN STAR 159
Exa 4.19 TO CALCULATE GENERATED EMF OF 3 PHASE STAR CONNECTED ALTERNATOR 160
Exa 5.1 TO DETERMINE EMF AND REGULATION AT A CERTAIN LOAD 162
Exa 5.2 TO DETERMINE PERCENTAGE REGULATION AT FULL LOAD LEADING AND LAGGING PF 163
Exa 5.3 TO DETERMINE PERCENTAGE REGULATION ON FULL LOAD 164
Exa 5.4 TO CALCULATE FULL LOAD REGULATION AT A LAGGING POWER FACTOR 164
Exa 5.5 TO FIND PERCENTAGE REGULATION AT CERTAIN LEADING AND LAGGING POWER FACTORS 166
Exa 5.6 TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD 167
Exa 5.7 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS 168
Exa 5.8 TO DETERMINE PERCENTAGE REGULATION AT CERTAIN LAGGING POWER FACTOR 171
Exa 5.9 TO DETERMINE FULL LOAD REGULATION AT VARIOUS POWER FACTORS 172
Exa 5.10 TO CALCULATE PERCENTAGE REGULATION FOR HALF LOAD 173
Exa 5.11 TO DETERMINE RATED TERMINAL VOLTAGE AND kVA RATING OF ALTERNATOR 175
Exa 5.12 TO DETERMINE INDUCED EMF AND TERMINAL VOLTAGE PER PHASE 176
Exa 5.13 TO DETERMINE VOLTAGE REGULATION BY EMF METHOD AT VARIOUS POWER FACTORS 177
Exa 5.14 TO FIND FULLLOAD VOLTAGE REGULATION USING SYNCHRONOUS IMPEDANCE METHOD . . 178
Exa 5.15 TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD 179

Exa 5.16 TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD 180
Exa 5.17 TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE . 182
Exa 5.18 TO FIND VOLTAGE REGULATION OF ALTERNATOR FOR FULL LOAD CURRENT USING POTIER METHOD 184
Exa 5.19 TO DETERMINE TERMINAL VOLTAGE AT A GIVEN EXCITATION 187
Exa 5.20 TO DETERMINE TERMINAL VOLTAGE LOAD ANGLE AND VOLTAGE REGULATION 188
Exa 5.21 TO DETERMINE VOLTAGE REGULATION BY EMF METHOD AT VARIOUS POWER FATORS 189
Exa 5.22 TO DETERMINE CERTAIN QUANTITIES ASSOCIATED WITH SINGLE PHASE ALTERNATOR . . . 191
Exa 5.23 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTOR 192
Exa 5.24 TO CALCULATE PERCENTAGE REGULATION AT LEADING LAGGING AND UNITY POWER FACTORS 193
Exa 5.26 TO CALCULATE PERCENTAGE REGULATION USING EMF METHOD 194
Exa 5.27 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO STAR CONNECTED ALTERNATOR . . 195
Exa 5.28 TO DETERMINE FULL LOAD PERCENTAGE REGULATION AT A LEADING AND LAGGING POWER FACTOR 196
Exa 5.29 TO CALCULATE PERCENTAGE REGULATION WHEN RATED OUTPUT SWITCHES OFF 198
Exa 5.30 TO CALCULATE VOLTAGE REGULATION FOR FULL LOAD CURRENT AT CERTAIN LEADING AND LAGGING POWER FACTORS 199
Exa 6.2 TO DETERMINE TOTAL INDUCED EMF ON OPEN CIRCUIT 201
Exa 6.3 TO DETERMINE CERTAIN CHARACTERISTICS OF SINGLE PHASE ALTERNATORS WORKING IN PARALLEL 202

Exa 6.4 TO CALCULATE SYNCHRONISING POWER OF ARMATURE PER MECHANICAL DEGREE OF PHASE DISPLACEMENT 203
Exa 6.5 CALCULATE SYNCHRONISING POWER AND TORQUE AT NO LOAD AND FULL LOAD 204
Exa 6.6 TO DETERMINE PERCENTAGE CHANGE IN INDUCED EMF REQUIRED TO BRING UNITY POWER FACTOR 205
Exa 6.7 TO DETERMINE LOAD SHARING AND UPF MAXIMUM LOAD 207
Exa 6.8 TO DETERMINE ARMATURE CURRENT OF ALTERNATOR 2 AND PF OF EACH ALTERNATORS 208
Exa 6.9 TO DETERMINE LOAD ON EACH MACHINE . . . 210
Exa 6.10 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMNT AND CORRESPONDING SYNCHRONISING TORQUE . 212
Exa 6.11 TO CALCULATE SYNCHRONISING POWER AND SYNCHRONISING TORQUE 213
Exa 6.12 TO CALCULATE SYNCHRONISING POWER AND CORRESPONDING SYNCHRONISING TORQUE . 214
Exa 6.13 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT . . 215
Exa 6.14 TO CALCULATE SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT . . 217
Exa 6.15 TO CALCULATE SYNCHRONISING POWER SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT 218
Exa 6.16 TO CALCULATE SYNCHRONIZING POWER PER MECHANICAL DEGREE OF DISPLACEMENT AND CORRESPONDING SYNCHRONIZING TORQUE . 221
Exa 6.17 DETERMINE THE LOAD SHARED BY EACH OF THE 2 MACHINES 222
Exa 6.18 TO DETERMINE THE EXCITATION OF 2ND ALTERNATORS 223
Exa 6.19 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT UNDER NOLOAD 225

Exa 6.20 TO FIND EMF AND POWER ANGLE 226
Exa 6.21 TO FIND THE EXCITATION EMF 228
Exa 6.22 TO DETERMINE REGULATION AND EXCITATION EMF REQUIRED TO MAINTAIN CERTAIN TERMINAL VOLTAGE 229
Exa 6.23 TO CALCULATE PERCENTAGE REGULATION OF THE MACHINE 229
Exa 6.24 TO CALCULATE PERCENTAGE VOLTAGE REGULATION AT A CERTAIN PF 230
Exa 6.25 TO DETERMINE LOAD ANGLE AND COMPONENTS OF ARMATURE CURRENT 231
Exa 6.26 TO COMPUTE PERCENTAGE REGULATION AT DIFFERENT POWER FACTOR 232
Exa 6.27 TO CALCULATE THE OUTPUT POWER FACTOR OF SECOND ALTERNATOR 233
Exa 6.28 TO CALCULATE THE POWER FACTOR OF SECOND MACHINE WORKING PARALLEL TO THE FIRST MACHINE 234
Exa 6.29 TO DETERMINE VOLTAGE REGULATION AND OPEN CIRCUIT POWER SUPPLY OF GENERATOR . . . 235
Exa 6.30 TO CALCULATE SYNCHRONISING POWER AND TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT 235
Exa 6.31 TO CALCULATE SYNCHRONIZING POWER PER MECHANICAL DEGREE OF DISPLACEMENT AND CORRESPONDING SYNCHRONISING TORQUE . 236
Exa 6.32 TO DETERMINE SYNCHRONOUS POWER PER MECHANICAL DEGREE OF DISPLACEMENT AT FULL LOAD 239
Exa 6.33 TO DETERMINE CERTAIN CHARACTERISTICS OF TWO ALTERNATORS OPERATING IN PARALLEL 240
Exa 6.34 TO DETERMINE OPEN CIRCUIT VOLTAGE . . . 241
Exa 6.35 FIND OUTPUT PF AND ARMATURE CURRENT OF SECOND MACHINE OPERATING IN PARALLEL WITH FIRST ALTERNATOR 241
Exa 6.36 TO DETERMINE ALTERED CURRENT AND POWER FACTOR 243

Exa 6.37 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO THREE PHASE STAR CONNECTED ALTERNATORS OPERATING IN PARALLEL 244
Exa 6.38 TO DETERMINE THE kW OUTPUT AND POWER FACTOR OF EACH OF THE SYNCHRONOUS GENERATOR 245
Exa 6.39 TO CALCULATE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT . . 247
Exa 6.40 TO DETERMINE THE ALTERNATOR CURRENT AND POWER FACTOR 248
Exa 6.41 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO EACH OF THE 2 ALTERNATORS . . . 250
Exa 6.42 TO CALCULATE THE EXCITATION VOLTAGE . . 251
Exa 6.43 TO DETERMINE EXCITATION EMF AT CERTAIN POWER FACTOR AND MAXIMUM LOAD THE MOTOR CAN SUPPLY AT NO EXCITATION 252
Exa 7.1 TO CALCULATE THE BACK EMF INDUCED IN THE MOTOR FOR VARIOUS POWER FACTORS . 254
Exa 7.2 TO DETERMINE THE OPERATING POWER FACTOR FOR DIFFERENT GENERATED EMF 255
Exa 7.3 TO DETERMINE GENERATED EMF ON FULL LOAD AND THE LOAD ANGLE 256
Exa 7.4 TO DETERMINE CURRENT DRAWN BY THE MOTOR AND ITS FULL LOAD EFFICIENCY 257
Exa 7.5 TO DETERMINE kVA RATING OFDESIRED SYNCHRONOUS MOTOR AND ITS OPERATING POWER FACTOR 258
Exa 7.6 TO DETERMINE INDUCED EMF ON FULL LOAD 260
Exa 7.7 TO CALCULATE MOTOR POWER FACTOR AND CURRENT DRAWN BY IT 261
Exa 7.8 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER 262
Exa 7.9 TO DETERMINE EMF AND MECHANICAL POWER DEVELOPED 264
Exa 7.10 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE MESH CONNECTED SYNCHRONOUS MOTOR 265

Exa 7.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 267
Exa 7.12 TO DETERMINE LOAD ANGLE ARMATURE CURRENT AND PF WHEN EXCITATION IS CHANGED 268
Exa 7.13 TO CALCULATE CURRENT AND PF IF INDUCED EMF IN SYNCHRONOUS MOTOR GETS INCREASED 270
Exa 7.14 TO FIND kVA RATING OF SYNCORONOUS MOTOR 271
Exa 7.15 TO FIND GROSS TORQUE DEVELOPED AND PF WITH CHANGING CURRENT AND LOAD TORQUE 271
Exa 7.16 TO DETERMINE ARMATURE CURRENT AND PF OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 273
Exa 7.17 TO CALCULATE ARMATURE CURRENT DRAWN BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 274
Exa 7.18 TO CALCULATE PF LOAD ANGLE AND ARMATURE CURRENT OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 276
Exa 7.19 TO FIND POWER FACTOR WHEN INPUT IS INCREASED 277
Exa 7.20 TO DETERMINE EMF GENERATED BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR . . 279
Exa 7.21 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER OF SYNCHRONOUS MOTOR 279
Exa 7.22 TO DETERMINE kVA INPUT TO SYNCHRONOUS MOTOR AND ITS POWER FACTOR WHEN DRIVING 6 kW LOAD 281
Exa 7.23 TO DETERMINE MINIMUM CURRENT AND INDUCED EMF AT FULL LOAD 282
Exa 7.24 TO DETERMINE PF WHEN INPUT OF SYNCHRONOUS MOTOR IS INCREASED 282
Exa 7.25 TO DETERMINE CURRENT AND PF OF A 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR . . 284
Exa 7.26 TO DETERMINE THE kVA RATING OF SYNCHRONOUS CONDENSER USED TO IMPROVE THE PF AND THE FACTORY 285

Exa 7.27 TO CALCULATE kVA INPUT AND PF OF SYNCHRONOUS MOTOR AT A CERTAIN INSTANT . . 286
Exa 7.28 TO DETERMINE MAXIMUM OUTPUT POWER OF SYNCHRONOUS MOTOR 288
Exa 7.29 TO DETERMINE INPUT POWER AND INDUCED EMF AT TWO DIFFERENT POWER FACTORS . . 288
Exa 7.30 TO DETERMINE AT FULLLOAD THE MINIMUM CURRENT AND ITS CORRESPONDING EMF . . . 289
Exa 7.31 TO DETERMINE MAXIMUM POWER AND TORQUE A THREE PHASE SYNCHRONOUS MOTOR CAN DELIVER 290

List of Figures

1.1 TO DRAW A DEVELOPED DIAGRAM FOR GENERATOR 8
1.2 TO DRAW DEVELOPED DIAGRAM FOR A DC GENERATOR . . . 11
1.3 TO FIND INDUCED EMF IN A GENERATOR . . . 15
1.4 TO DETERMINE ARMATURE RESISTANCE OF GENERATOR . . . 16
1.5 TO DETERMINE TERMINAL VOLTAGE AT THE LOAD 18
1.6 TO CALCULATE THE VOLTAGE GENERATED BY SHUNT COMPOUND DC GENERATOR . . . 19
1.7 TO CALCULATE THE OPEN CIRCUIT VOLTAGE AND LOAD CURRENT . . . 21
1.8 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR USING ITS MAGNETISING CURVE 23
1.9 TO CALCULATE LOAD CURRENT . . . 25
1.10 TO CALCULATE ARMATURE CURRENT AND GENERATED EMF . . . 25
1.11 TO DETERMINE THE TERMINAL VOLTAGE . . . 27
1.12 TO CALCULATE CERTAIN QUANTITIES FROM OPEN CIRCUIT CHARACTERISTICS OF DC SHUNT MOTOR 29
1.13 TO DESIGN A LAP WINDING . . . 32
1.14 TO DRAW DEVELOPED ARMATURE WINDING DIAGRAM OF DC MACHINE . . . 36
1.15 TO CALCULATE ARMATURE CURRENT AND OUTPUT POWER . . . 40
1.16 TO DETERMINE ALTERED CURRENT WHEN SPEED OF SEPERATELY EXCITED GENERATOR IS DROPPED 46

2.1 TO CALCULATE SPEED WHEN MOTOR DRAWS 60 A FROM SUPPLY . . . 51

2.2 TO DETERMINE ARMATURE CURRENT AND BACK EMF 52
2.3 TO DETERMINE SPEED ON FULL LOAD 54
2.4 TO DETERMINE SPEED OF MOTOR WITH ALTERED LOAD 56
2.5 TO DETERMINE ARMATURE CURRENT WHEN A RESISTANCE IS ADDED IN SERIES TO FIELD WINDING . 57
2.6 TO DETERMINE MOTOR SPEED WHEN FIELD WINDING GETS SHUNTED BY A RESISTANCE 58
2.7 TO DETERMINE EXTRA RESISTANCE THAT WILL REDUCE THE SPEED 60
2.8 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR 63
2.9 TO DETERMINE MECHANICAL POWER AND NOLOAD SPEED AND CURRENT 64
2.10 TO DETERMINE SPEED IF FIELD WINDING IS SHUNTED BY ADDITIONAL RESISTANCE 68
2.11 TO DETERMINE SPEED IF FIELD GROUPS ARE ARRANGED IN PARALLEL 70
2.12 TO DETERMINE NEW SPEED AND ARMATURE CURRENT AFTER RECONNECTION 72
2.13 TO CALCULATE EXTRA RESISTANCE TO REDUCE THE SPEED 75
2.14 TO DETERMINE ADDITIONAL RESISTANCE IN FIELD CIRCUIT TO RAISE THE SPEED 76
2.15 TO CALCULATE RESISTANCE TO BE CONNECTED IN SERIES WITH ARMATURE TO HALVE THE SPEED . . 78
2.16 TO CALCULATE EXTRA RESISTANCE IN SERIES WITH ARMATURE TO REDUCE SPEED AT FULL LOAD . . . 80
2.17 TO DETERMINE SPEED AND TORQUE DEVELOPED AT FULL LOAD WHEN NO LOAD FLUX WEAKENS . . 82
2.18 TO FIND THE SPEED WHEN ADDITIONAL RESISTANCES GET CONNECTED WITH SHUNT FIELD AND ARMATURE 83
2.19 TO DETERMINE EXTRA RESISTANCE WITH FIELD CURRENT TO INCREASE SPEED OF DC SHUNT MOTOR . 84
2.20 TO FIND EXTRA RESISTANCE TO BE ADDED IN SERIES WITH ARMATURE TO REDUCE ITS SPEED WITH SAME ARMATURE CURRENT 85

2.21 TO DETERMINE THE SPEED WHEN ADDITIONAL RESISTANCE GETS CONNECTED AND DRAWING SAME CURRENT 87
2.22 TO DETERMINE ADDITIONAL RESISTANCE IN SERIES WITH ARMATURE TO REDUCE THE SPEED AND ALTERED SPEED WHEN TORQUE GETS HALVED 88
2.23 TO DETERMINE RESISTANCE TO BE INSERTED IN SHUNT FIELD CIRCUIT TO INCREASE THE SPEED 91

3.1 TO DETERMINE THE EFFICIENCY OF MACHINES . . 101
3.2 TO FIND EFFICIENCY OF EACH MACHINE 103
3.3 TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V DC SHUNT MOTOR 117
3.4 TO DETERMINE EFFICIENCY OF EACH OF THE 2 SHUNT MACHINES 122
3.5 TO DETERMINE EFFICIENCY OF MOTOR AND GENERATOR 124
3.6 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES 126
3.7 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR ON FULL LOAD 128
3.8 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES 130
3.9 TO CALCULATE EFFICIENCY OF MACHINE ACTING AS GENERATOR 131
3.10 TO CALCULATE EFFICIENCY OF DC MACHINES . . . 133
3.11 TO ESTIMATE EFFICIENCY OF 2 DC MACHINES . . . 135
3.12 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR 137
3.13 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR 139

4.1 TO DRAW THE DIAGRAM FOR FULL PITCH ARMATURE WINDING OF AN ALTERNATOR 142

5.1 TO CALCULATE FULL LOAD REGULATION AT A LAGGING POWER FACTOR 165

5.2 TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD . . . 169
5.3 TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD . . . 169
5.4 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS 171
5.5 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS 172
5.6 TO CALCULATE PERCENTAGE REGULATION FOR HALF LOAD . . . 174
5.7 TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD 181
5.8 TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD 181
5.9 TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD . . . 183
5.10 TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD . . . 183
5.11 TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE 185
5.12 TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE 185
5.13 TO FIND VOLTAGE REGULATION OF ALTERNATOR FOR FULL LOAD CURRENT USING POTIER METHOD 186
5.14 TO DETERMINE TERMINAL VOLTAGE LOAD ANGLE AND VOLTAGE REGULATION . . . 188
5.15 TO DETERMINE FULL LOAD PERCENTAGE REGULATION AT A LEADING AND LAGGING POWER FACTOR 197

6.1 TO DETERMINE LOAD SHARING AND UPF MAXIMUM LOAD . . . 207
6.2 TO DETERMINE ARMATURE CURRENT OF ALTERNATOR 2 AND PF OF EACH ALTERNATORS 209
6.3 TO DETERMINE LOAD ON EACH MACHINE 211
6.4 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT 216

6.5 TO CALCULATE SYNCHRONISING POWER SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT 219
6.6 DETERMINE THE LOAD SHARED BY EACH OF THE 2 MACHINES 222
6.7 TO DETERMINE THE EXCITATION OF 2ND ALTERNATORS 224
6.8 TO FIND EMF AND POWER ANGLE 227
6.9 TO DETERMINE SYNCHRONOUS POWER PER MECHANICAL DEGREE OF DISPLACEMENT AT FULL LOAD . . 238
6.10 TO DETERMINE ALTERED CURRENT AND POWER FACTOR 243
6.11 TO DETERMINE THE kW OUTPUT AND POWER FACTOR OF EACH OF THE SYNCHRONOUS GENERATOR 246
6.12 TO DETERMINE THE ALTERNATOR CURRENT AND POWER FACTOR 249

7.1 TO DETERMINE GENERATED EMF ON FULL LOAD AND THE LOAD ANGLE 256
7.2 TO DETERMINE kVA RATING OFDESIRED SYNCHRONOUS MOTOR AND ITS OPERATING POWER FACTOR 259
7.3 TO CALCULATE MOTOR POWER FACTOR AND CURRENT DRAWN BY IT 261
7.4 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER 263
7.5 TO DETERMINE EMF AND MECHANICAL POWER DEVELOPED 264
7.6 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 267
7.7 TO DETERMINE LOAD ANGLE ARMATURE CURRENT AND PF WHEN EXCITATION IS CHANGED 269
7.8 TO CALCULATE CURRENT AND PF IF INDUCED EMF IN SYNCHRONOUS MOTOR GETS INCREASED 270
7.9 TO DETERMINE ARMATURE CURRENT AND PF OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR . 273
7.10 TO CALCULATE ARMATURE CURRENT DRAWN BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR . 275

7.11 TO CALCULATE PF LOAD ANGLE AND ARMATURE CURRENT OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 276
7.12 TO FIND POWER FACTOR WHEN INPUT IS INCREASED 278
7.13 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER OF SYNCHRONOUS MOTOR 280
7.14 TO DETERMINE CURRENT AND PF OF A 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR 284
7.15 TO DETERMINE MAXIMUM OUTPUT POWER OF SYNCHRONOUS MOTOR 287

Chapter 1

DC Generators

Scilab code Exa 1.1 TO DETERMINE EMF GENERATED DUE TO ROTATION AND REPLACEMENT OF LAP WOUND ARMATURE WITH WAVE WOUND

```
1 clc,clear
2 printf('Example 1.1\n\n')
3
4 Pole=4
5 Z=440   //number of conductors in armature
6 phi=0.07  //flux produced by each pole in webers
7 N=900     //Speed of armature in r.p.m
8
9 //Part(i)  lap wound
10 A1=Pole  //no of parallel paths for lap winding
11 E1=phi*N*Z*Pole/(60*A1)
12 printf('(i) e.m.f generated (lap-wound) is %.0f V',
     E1)
13
14 //Part(ii)  wave wound
15 A2=2  //no of parallel paths for wave winding
16 E2=phi*N*Z*Pole/(60*A2)
17 printf('\n(ii) e.m.f generated (wave-wound)is %.0f V
     ',E2)
```

Scilab code Exa 1.2 TO DETERMINE GENERATED EMF AND THE SPEED TO GENERATE THE SAME EMF USING WAVE WOUND ARMATURE

```
clc,clear
printf('Example 1.2\n\n')

Pole=4
phi=21*10^-3  //flux produced by each pole in webers
N=1120        //Speed of armature in r.p.m
Coils=42
turns_per_coil=8
Turns=Coils * turns_per_coil
Z=2*Turns  //Number of armature conductors

//Part(i)
A1=Pole  //no of parallel paths for lap winding
E1=phi*N*Z*Pole/(60*A1)
printf('(i)  e.m.f generated is %.3f V',E1)

//Part(ii)
A2=2  //wave winding
E2=E1 //as mentioned in the question
N2=E2/(phi*Z*Pole/(60*A2)) //E=phi*N*Z*Pole/(60*A)
printf('\n(ii)  For wave-wound armature,above
   calculated e.m.f is generated at %.0f r.p.m',N2)
```

Scilab code Exa 1.3 TO DRAW A DEVELOPED DIAGRAM FOR GENERATOR

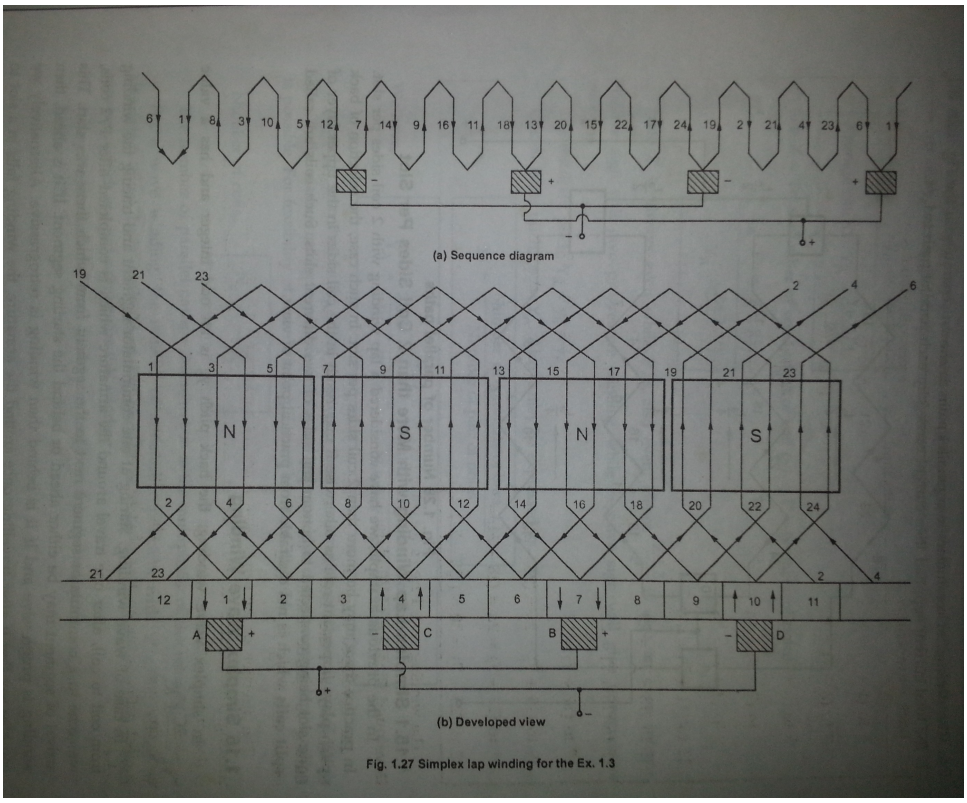


Fig. 1.27 Simplex lap winding for the Ex. 1.3

Figure 1.1: TO DRAW A DEVELOPED DIAGRAM FOR GENERATOR

```
clc,clear
printf('Example 1.3\n\n')

Pole=4
coils=12
commutator_segments=coils
coil_sides=coils*2
Z=coil_sides  //No of conductors
pole_pitch=Z/Pole

//for Simplex lap winding
y_f=pole_pitch-1
y_b=pole_pitch+1

y_c=1  //Note that it's positive and it's
   progressive type of Simplex lap winding


printf('WINDING TABLE:\n\n    1<-     8->     3<-
   10->     5<-    12\n-> 7<-    14->     9<-    16->
   11<-    18\n->13<-    20->    15<-    22->    17<-
   24\n->19<-     2->    21<-     4->    23<-     6\n->
   1\n

   ')
printf('\nNote that <- indicates back connection
   with  y_back=%.0f and -> indicates front
   connection with y_front=%.0f\n',y_b,y_f)
printf('\nAnother form of winding table:')
printf('\n       BACK CONNECTIONS
                                        FRONT CONNECTIONS'
   )

printf('\n\n        1 to (1+7) = 8
                          ->          8 to (8-5) = 3')
printf('\n        3 to (3+7) =10
                          ->         10 to (10-5)= 5')
printf('\n        5 to (5+7) =12
```

```
                                    ->          12 to (12-5)= 7')
26 printf('\n        7 to (7+7) =14
                                    ->          14 to (14-5)= 9')
27 printf('\n        9 to (9+7) =16
                                    ->          16 to (16-5)=11')
28 printf('\n       11 to (11+7)=18
                                    ->          18 to (18-5)=13')
29 printf('\n       13 to (13+7)=20
                                    ->          20 to (20-5)=15')
30 printf('\n       15 to (15+7)=22
                                    ->          22 to (22-5)=17')
31 printf('\n       17 to (17+7)=24
                                    ->          24 to (24-5)=19')
32 printf('\n       19 to (19+7)=26=(26-24)=2
                   ->          2 to (26-5)=21')
33 printf('\n       21 to (21+7)=28=(28-24)=4
                   ->          4 to (28-5)=23')
34 printf('\n       23 to (23+7)=30=(30-24)=6
                   ->          6 to (30-5)=25  = 25-24=1'
   )
```

Scilab code Exa 1.4 TO DRAW DEVELOPED DIAGRAM FOR A DC GENERATOR

```
1 clc,clear
2 printf('Example 1.4\n\n')
3
4 Pole=4
5 Z=18 //no of armature conductors
6 Y_A=(Z+2)/Pole  //For progressive type wave winding,
     positive sign is used
7 Y_C=Y_A  //For wave winding
8
```

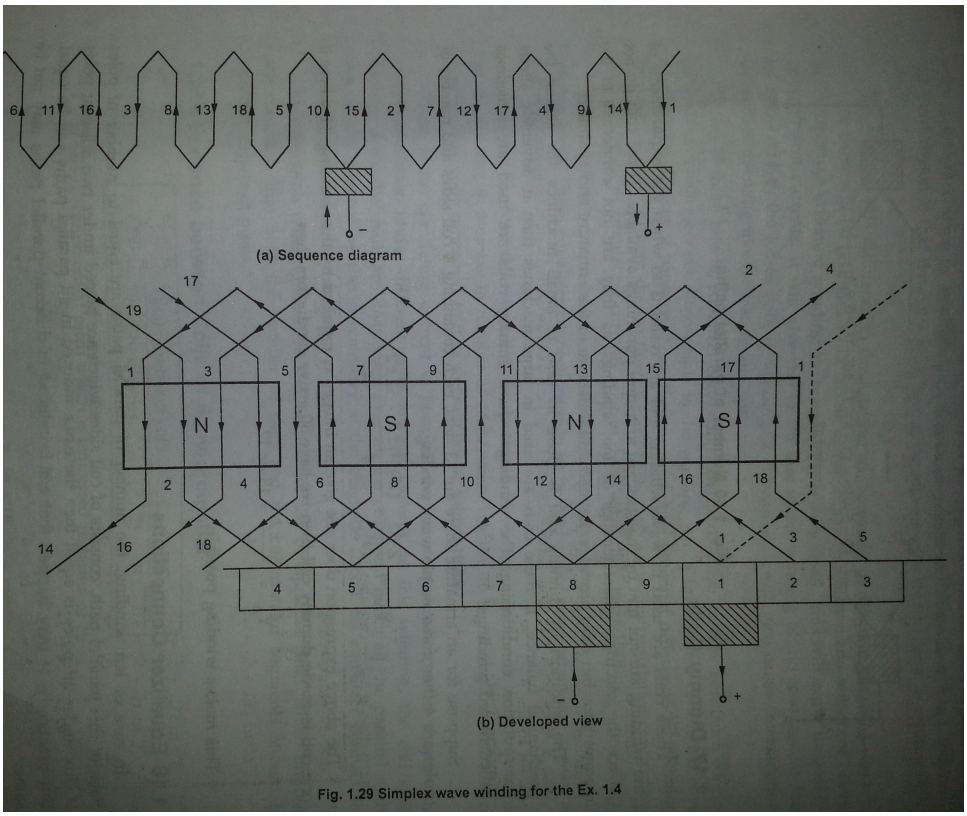


Figure 1.2: TO DRAW DEVELOPED DIAGRAM FOR A DC GENERATOR

```
9  //Since Y_A=(y_b+y_f)/2, we let y_b=Y_f
10 y_b=Y_A/2    //say
11 y_f=y_b
12
13 coils=Z/2
14 slots=coils
15 commutator_segments=coils
16
17 printf('WINDING TABLE:\n    1<-     6->   11<-   16->
      3<-    8\n->13<-    18->    5<-   10->   15<-    2\n->
      7<-    12->   17<-    4->    9<-   14\n->1\n

      ')
18
19 printf('\nAnother form of winding table:')
20
21 printf('\n       BACK CONNECTIONS
                                          FRONT CONNECTIONS'
      )
22
23 printf('\n\n         1 to (1+5)  = 6
                          ->          6 to (6+5) = 11')
24 printf('\n        11 to (11+5) =16
                          ->         16 to (16+5)= 21
      -18=3')
25 printf('\n         3 to (3+5)  = 8
                          ->          8 to  (8+5)= 13')
26 printf('\n        13 to (13+5) =18
                          ->         18 to (18+5)= 23
      -18=5')
27 printf('\n         5 to (5+5)  =10
                          ->         10 to (10+5)= 15')
28 printf('\n        15 to (15+5) =20 -18=2
                        ->          2 to  (2+5)=  7')
29 printf('\n         7 to (7+5)  =12
                          ->         12 to (12+5)= 17')
30 printf('\n        17 to (17+5) =22 -18=4
                        ->          4 to  (4+5)=  9')
```

```
31 printf('\n        9 to (9+5)   =14
                              ->          14 to (14+5)= 19
   -18=1')
```

Scilab code Exa 1.5 TO CALCULATE DEMAGNETISING AND CROSS-MAGNETISING AMPERE TURNS PER POLE

```
1  clc,clear
2  printf('Example 1.5\n\n')
3
4  Pole=4
5  Z=480 //No of armature conductors
6  I_a=144
7  I=I_a/2  //For wave wound
8  theta_m=10  //lead angle in DEGREES
9
10 amp_turns_PP_d=Z*I*theta_m/360  //demagnetising
     Ampere-turns per pole
11 amp_turns_PP_c=Z*I*(1/(2*Pole)-theta_m/360) //cross-
     magnetising Ampere-turns per pole
12
13 printf('De-magnetising ampere-turns per pole is %.0f
     ',amp_turns_PP_d)
14 printf('\nCross-magnetising ampere-turns per pole is
     %.0f',amp_turns_PP_c)
```

Scilab code Exa 1.6 TO DETERMINE NUMBER OF COMPENSATING CONDUCTORS PER POLE

```
1  clc,clear
2  printf('Example 1.6\n\n')
3
4  Pole=10
```

```
5 Z=800 //No of armature conductors
6 A=Pole //For lap wound
7 ratio=0.7  //ratio of pole arc to pole pitch
8 //amp_turns_PP=ratio*(I_a*Z)/(2*A*P)
9 turns_PP=ratio*(Z)/(2*A*Pole)   //turns per pole
10 conductors_PP=turns_PP*2 //multiplied with 2 because
     2 conductors form 1 turn
11
12 printf('Compensating conductors per pole= %.0f',ceil
     (conductors_PP))
```

Scilab code Exa 1.7 TO FIND REACTIVE VOLTAGE DURING LINEAR AND SINUSOIDAL COMMUTATION

```
1 clc,clear
2 printf('Example 1.7\n\n')
3
4 I_L=150,A=4
5 N=1800 //in rpm
6 W_b=1.2  //Brush width
7 W_m=0  //width of mica insulation
8 L=0.06*10^-3 //Inductance
9 segments=64
10 n_s=1800/60   //in rps and not rpm
11 v=n_s*segments  //peripheral speed in segments per
     second
12
13 T_c=(W_b-W_m)/v  //Time of commutation
14 I=I_L/A  //Current through a conductor
15
16 //Part(i)
17 E_l=L*2*I/T_c
18 printf('\n(i)   Reactive voltage using Linear
     commutation is %.1f V',E_l)
19
```

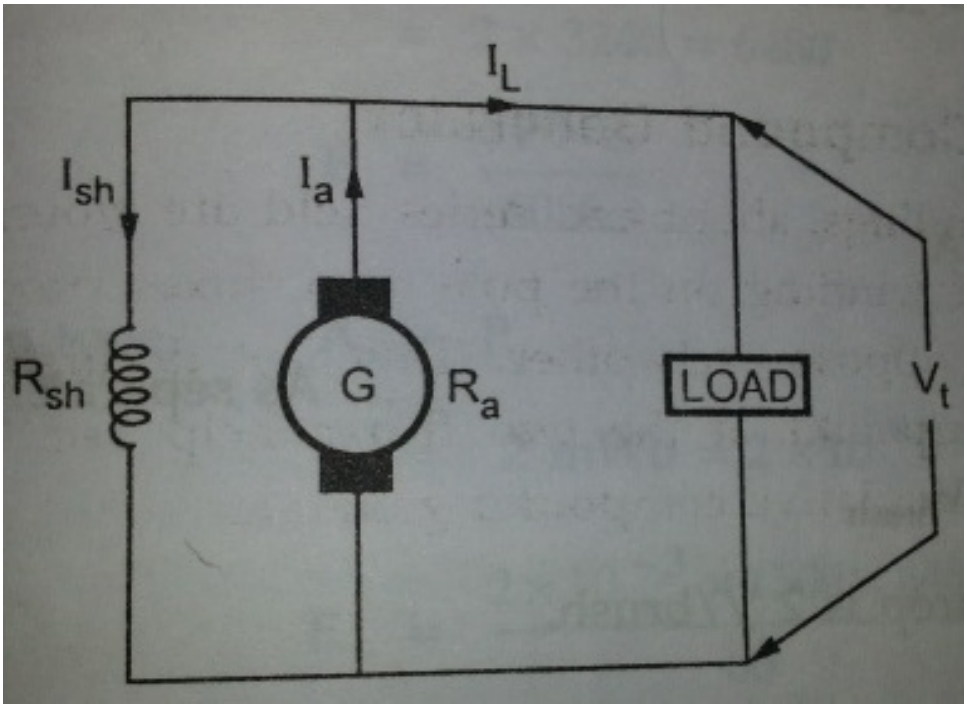


Figure 1.3: TO FIND INDUCED EMF IN A GENERATOR

```
20 //Part(ii)
21 E_s=1.11*L*2*I/T_c
22 printf('\n(ii)  Reactive voltage using Sinusoidal
      commutation is %.3f V',E_s)
```

Scilab code Exa 1.8 TO FIND INDUCED EMF IN A GENERATOR

```
1 clc,clear
2 printf('Example 1.8\n\n')
3
4 V_t=250  //Terminal voltage
5 R_sh=100  //Resistance of shunt field winding
```

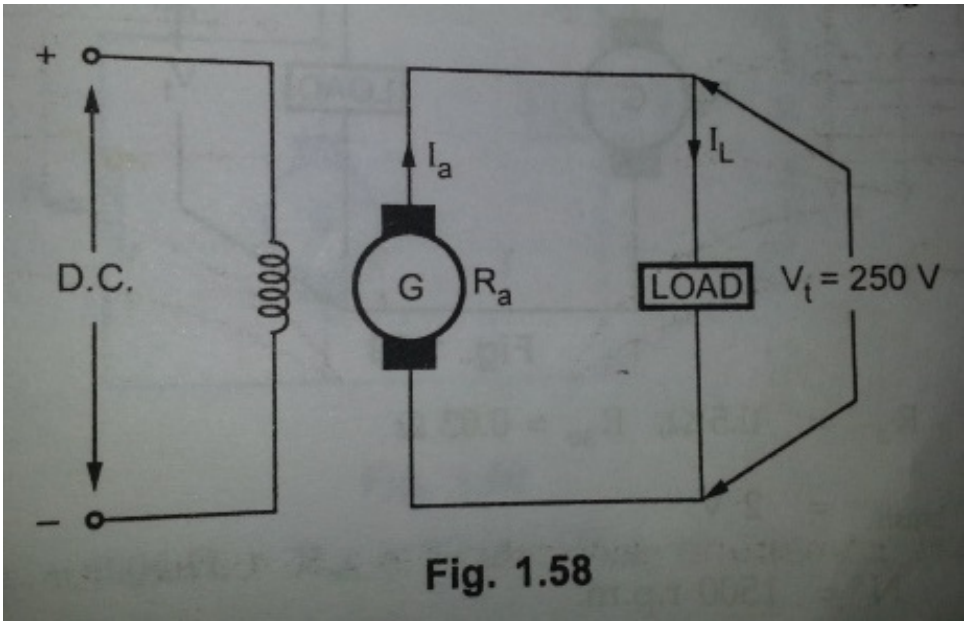


Figure 1.4: TO DETERMINE ARMATURE RESISTANCE OF GENERATOR

```
6  I_sh=V_t/R_sh //shunt current
7  R_a=0.22  //Armature resistance
8
9  P=5*10^3  //Load power
10 I_L=P/V_t  //Load current
11 I_a=I_L+I_sh //armature current
12
13 E=V_t + I_a*R_a  //Induced emf
14 printf('\nInduced e.m.f to supply the 5kW load is %
   .2f V',E)
```

Scilab code Exa 1.9 TO DETERMINE ARMATURE RESISTANCE OF GENERATOR

```
clc,clear
printf('Example 1.9\n\n')

V_t=250  //terminal voltage
P=10*10^3  //10kW power of generator
I_L=P/V_t //load current
I_a=I_L  //As seperately excited
V_brush=2*2 // 2 * no of brushes

E=255  //on full load
R_a=(E-V_t-V_brush)/I_a //Because E=V_t+ I_a*R_a  + V_brush

printf('\nArmature resistance of generator is %.3f ohm',R_a)
```

Scilab code Exa 1.10 TO DETERMINE TERMINAL VOLTAGE AT THE LOAD

```
clc,clear
printf('Example 1.10\n\n')

R_a=0.5,R_se=0.03  //resitance due to armature and series field winding
V_brush=2 //brush drop
N=1500 //generator speed in r.p.m
coils=540
turns_per_coil=6
total_turns= coils*turns_per_coil
Z=2*total_turns //Total conductors
I_a=50 //armature current

phi=2*10^-3  //flux per pole in webers
```

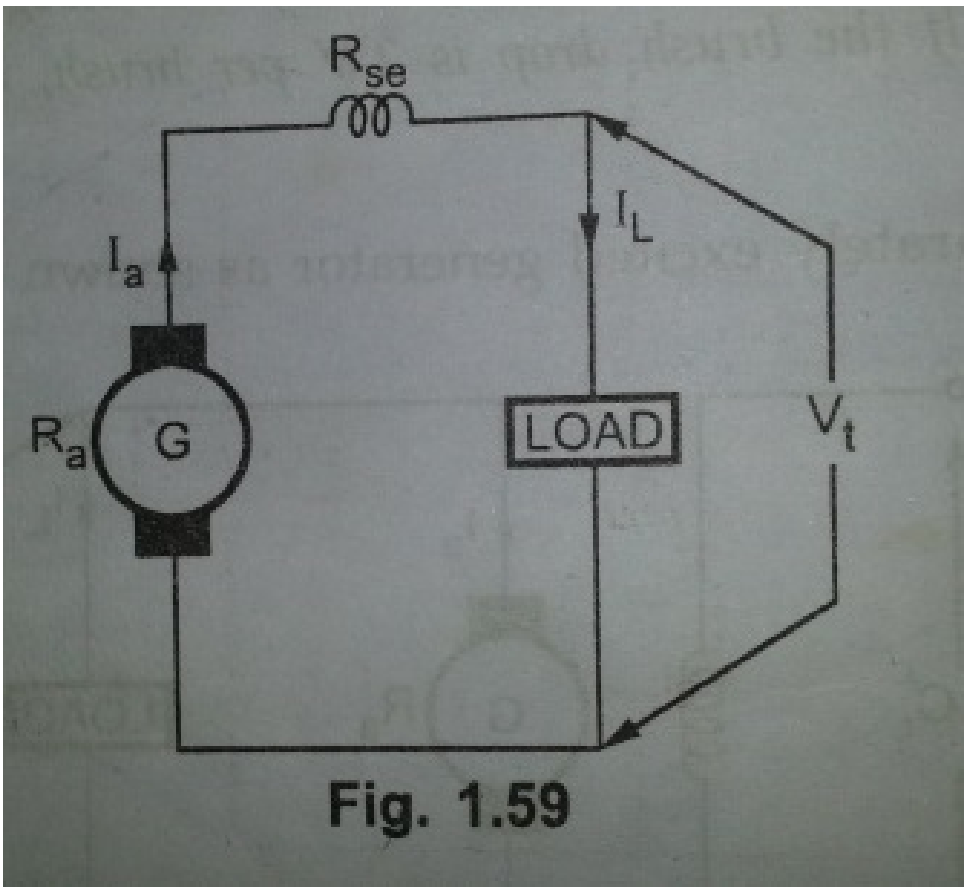


Figure 1.5: TO DETERMINE TERMINAL VOLTAGE AT THE LOAD

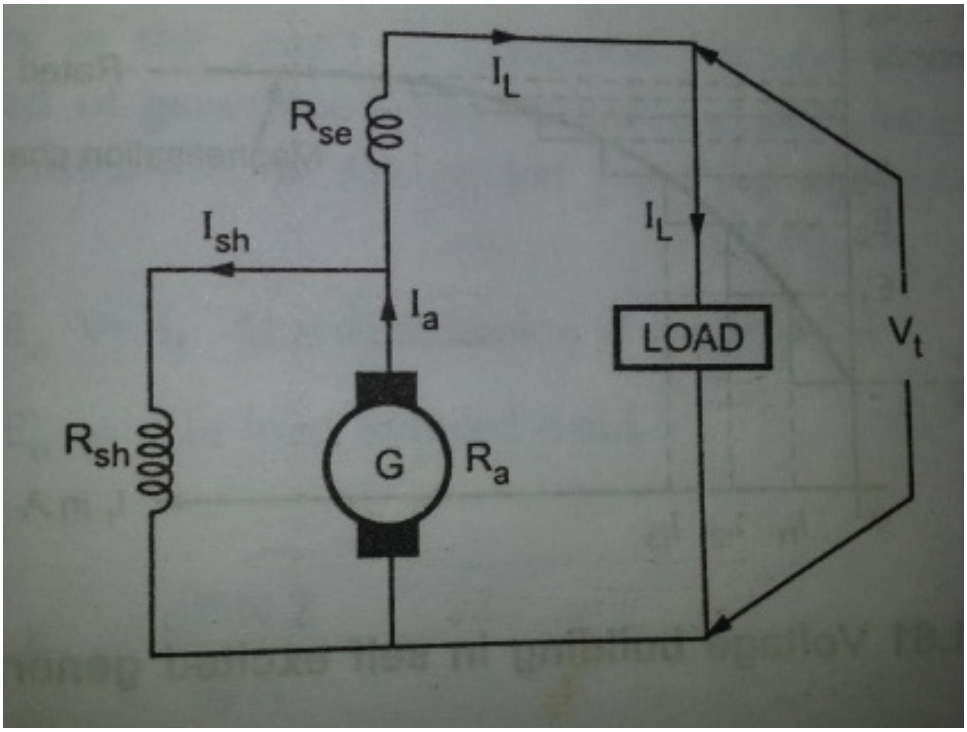


Figure 1.6: TO CALCULATE THE VOLTAGE GENERATED BY SHUNT COMPOUND DC GENERATOR

```
14 E=phi*N*Z/(60)  //A=P for lap-wound and they cancel
       out
15 V_t =E- (I_a*(R_a+R_se)  + V_brush)   //Because E=
       V_t+ I_a*R_a  + V_brush
16 printf('\nTerminal voltage is %.1f V',V_t)
```

Scilab code Exa 1.11 TO CALCULATE THE VOLTAGE GENERATED BY SHUNT COMPOUND DC GENERATOR

```
1 clc,clear
```

```
2 printf('Example 1.11\n\n')
3
4 V_t=225 //voltage across winding
5 R_a=0.04 //armature resistance
6 R_sh=90 //shunt resistance
7 R_se=0.02//resistance of series field winding
8 I_L=75 //load current
9
10 //E -I_a*R_a=V_t+I_L*R_se
11 I_sh=(V_t+I_L*R_se)/R_sh  //current through shunt
     field winding
12
13 I_a=I_L + I_sh  //armature current
14 E=V_t+ I_a*R_a+I_L*R_se  //induced emf
15
16 printf('\nGenerated voltage is %.1f V',E)
```

Scilab code Exa 1.12 TO CALCULATE THE OPEN CIRCUIT VOLTAGE AND LOAD CURRENT

```
1 clc,clear
2 printf('Example 1.12\n\n')
3
4 R_sh=53  //Resistance of field winding
5 V_t=100 //terminal voltage
6 I_sh =V_t/R_sh //shunt current
7 I_f=I_sh
8 R_a=0.1 //armature resistance
9 E_o=143 // for I_sh= I_f = 1.8867 as obtained from
     graph
10 I_a=(E_o-V_t)/R_a  //Because E_o=V_t + I_a*R_a
11 I_L=I_a-I_sh  //no load current
12 printf('\n\nNote: Open circuit voltage was obtained
```

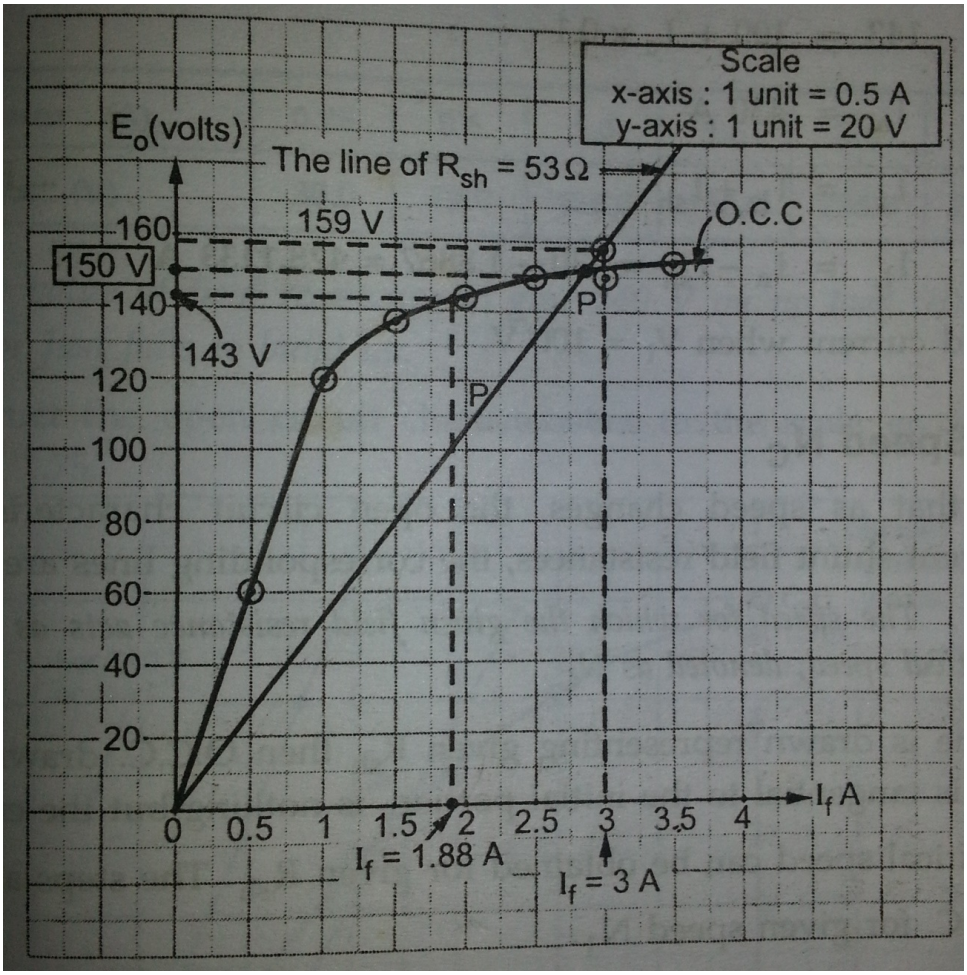


Figure 1.7: TO CALCULATE THE OPEN CIRCUIT VOLTAGE AND LOAD CURRENT

```
    as follows\nE_o=R_sh*I_f    // y=mx+c form with c
    =0 and R_sh=53\nHence, a line with slope 53
    through origin is made to intersect OCC at 150 V'
    )
13
14 printf('\nTherefore, Open circuit voltage is 150 V')
15 printf('\n\n\nNo load current is %.4f A    ',I_L)
```

Scilab code Exa 1.13 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR USING ITS MAGNETISING CURVE

```
1  clc,clear
2  printf('Example 1.13\n\n')
3
4  //part(1)
5  E_o=240  //on no-load
6
7  //Draw horizontal line from 240 V, to intersect OCC
     at A. corresponding I_f is 2.25 A
8  //The slope pf OA is corresponding R_sh
9  I_f=2.25  //Corresponds to 240 V when intersected
     OCC
10 R_sh=240/I_f  //shunt resistance
11 printf('(i)Field resistance that gives 240 V on no-
     load is %.2f ohms \n',R_sh)
12
13 //Part(ii)
14 N1=1000 //speed of shunt generator in rpm
15 I_f=1
16
17 //Draw line OP tangential to OCC at N1=1000 r.p.m.
18 //Select I_f=1A i.e. point R
19 //Draw vertical from R to intersect OP at S and OA
```

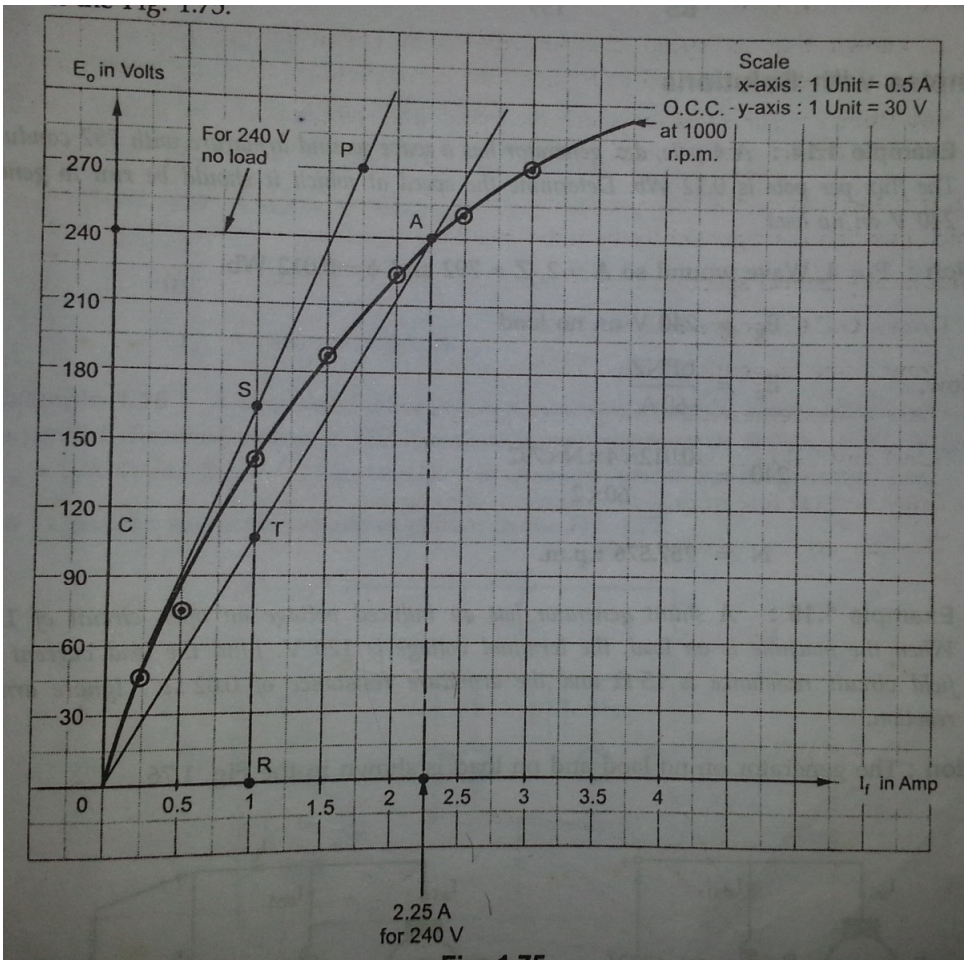


Figure 1.8: TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR USING ITS MAGNETISING CURVE

```
    at T....this gives RT=105 and RS=159
//At critcal speed generator just fails to build up

RT=105,RS=159
N_C=N1*RT/RS //Critical speed
printf('(ii)Critical speed is %.2f r.p.m ',N_C)
```

Scilab code Exa 1.14 TO DETERMINE RUNNING SPEED TO GENERATE 240 V ON NOLOAD

```
clc,clear
printf('Example 1.14\n\n')

P=4  //number of poles
A=2 //because wave wound
Z=792 //No of conductors
phi=0.012  //flux per pole in weber
E_g=240 //on no-load
//running speed
N=E_g*60*A/(phi*P*Z) //becuase E_g= phi*P*N*Z/(60*A)

printf('Required running speed is %.3f r.p.m',N)
```

Scilab code Exa 1.15 TO CALCULATE LOAD CURRENT

```
clc,clear
printf('Example 1.15\n\n')

//open circuit condition
I_L=0  //because of open circuit
V_t=127 //terminal voltage
```

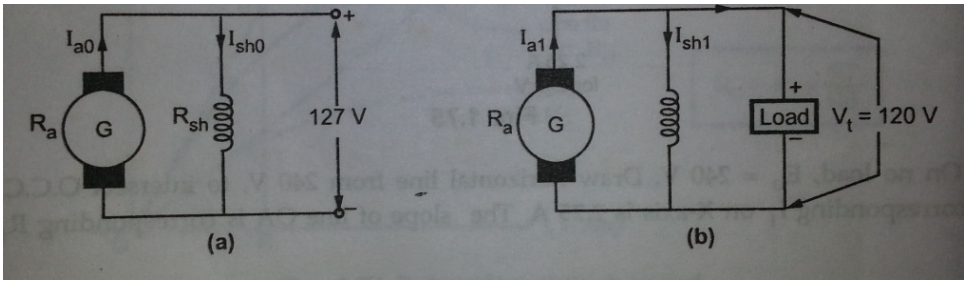
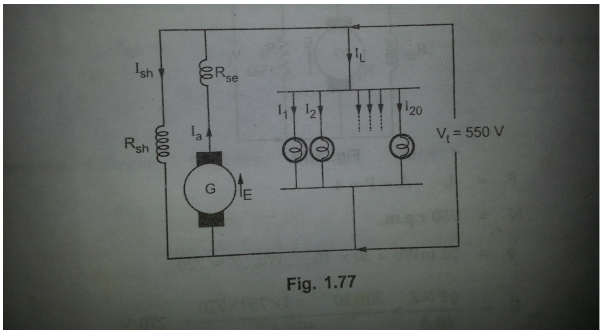
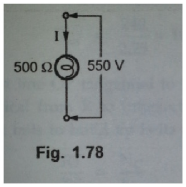


Figure 1.9: TO CALCULATE LOAD CURRENT





Considering a single lamp

Figure 1.10: TO CALCULATE ARMATURE CURRENT AND GENERATED EMF

```
7  E_g=V_t //because I_L=0
8
9  //load condition
10 V_t=120
11 R_sh=15,R_a=0.02  //shunt and armature resistance
12 I_sh1=V_t/R_sh //current through shunt winding in
      loaded condition
13
14 I_L =(E_g-V_t)/R_a  -  I_sh1 //because I_a1=I_L+
      I_sh1    and    E_g=V_t + I_a1*R_a
15 printf('Load current is %.0f A',I_L)
```

Scilab code Exa 1.16 TO CALCULATE ARMATURE CURRENT AND GENERATED EMF

```
clc,clear
printf('Example 1.16\n\n')

V_t=550  //Terminal voltage
R_lamp=500 //Each lamp
I_lamp=V_t/R_lamp  //each lamp ; V_t because all lamps are in parallel

I_L=20*I_lamp  //there exist 20 lamps
R_sh=25,R_a=0.06,R_se=0.04  //resistance of shunt winding,armature,series field
I_sh=V_t/R_sh //current throough shunt winding
I_a=I_L+I_sh//armature current
E=V_t + I_a*(R_a+R_se)  //generated emf

printf('Armature current and generated e.m.f is %.0f A and %.1f V respectively ',I_a,E )
```

Scilab code Exa 1.17 TO DETERMINE THE TERMINAL VOLTAGE

```
clc,clear
printf('Example 1.17\n\n')

P=4 //number of poles
A=P // because of lap wound
N=750 //speed in rpm
Z=720 //number of armature conductors
```

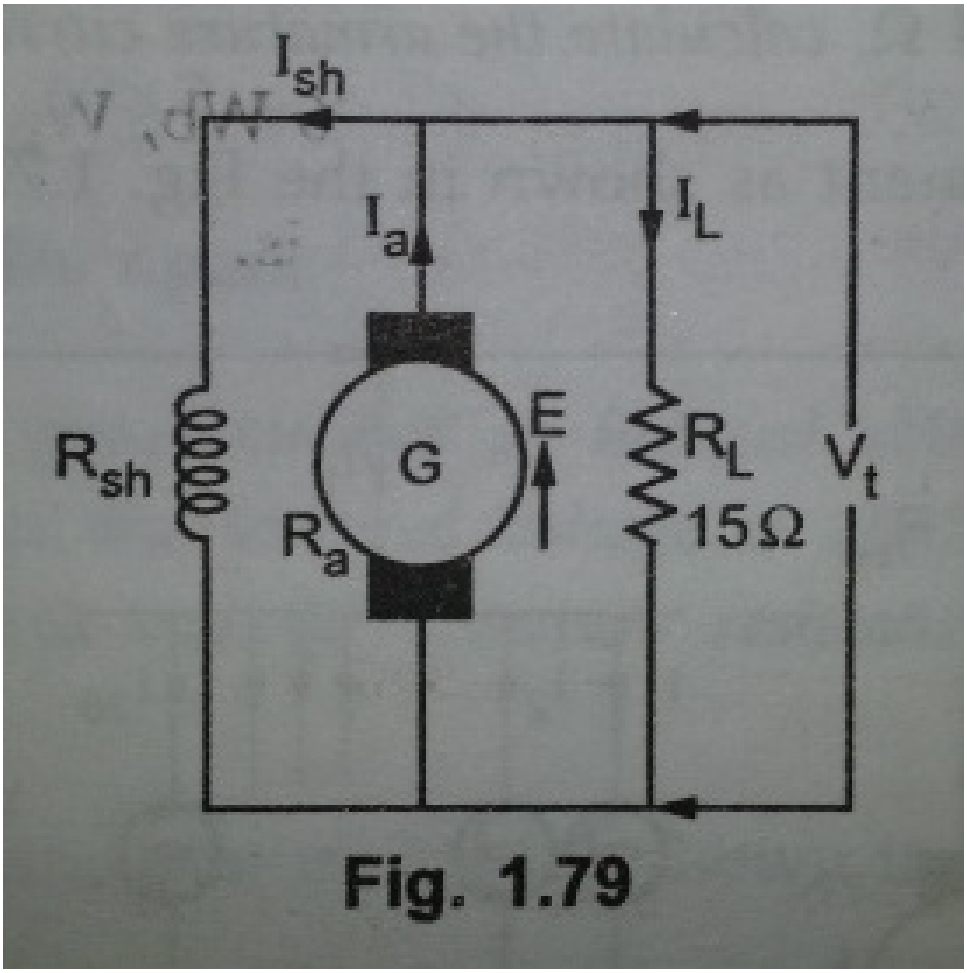


Figure 1.11: TO DETERMINE THE TERMINAL VOLTAGE

```
8  phi=30*10^-3  //flux per pole in weber
9  R_sh=200,R_a=0.4,R_L=15, //resistance of shunt
       winding,armature,series field
10 E=phi*P*N*Z/(60*A) //generated emf
11
12 //solving the following equations for V_t
13 //E= V_t + I_a*R_a
14 //E= V_t + (I_L + I_sh)*R_a
15 //E= V_t + ((V_t/R_L) + (V_t/R_sh))*R_a
16 V_t=E/(1+(R_a/R_L)+(R_a/R_sh))
17 printf('Terminal voltage = %.4f V',V_t)
```

Scilab code Exa 1.18 TO DETERMINE THE DRIVING SPEED OF ARMATURE TO GENERATE CERTAIN EMF

```
1  clc,clear
2  printf('Example 1.18\n\n')
3
4  P=6 //number of poles
5  A=2 // because of wave wound
6  N_1=300 //speed of generator
7  Z=600 //number of armature conductors
8  phi_1=0.06  //flux per pole in webers
9  E_g1=phi_1*P*N_1*Z/(60*A) //generated emf
10 printf('Emf generated is %.0f V\n\n',E_g1)
11
12 phi_2=0.055 //new flux per pole
13 E_g2=550// new generated emf
14 N_2=E_g2/(phi_2*P*Z/(60*A)) //new speed of generator
15 printf('Required speed is %.2f r.p.m',N_2)
```

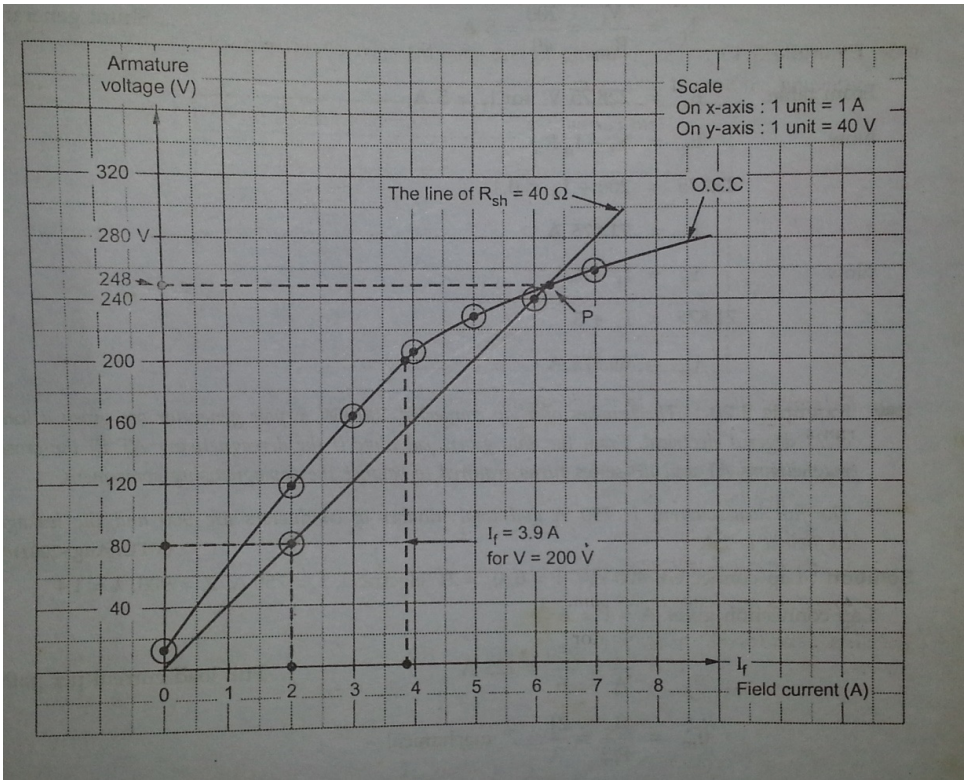


Figure 1.12: TO CALCULATE CERTAIN QUANTITIES FROM OPEN CIRCUIT CHARACTERISTICS OF DC SHUNT MOTOR

Scilab code Exa 1.19 TO CALCULATE CERTAIN QUANTITIES FROM OPEN CIRCUIT CHARACTERISTICS OF DC SHUNT MOTOR

```
clc,clear
printf('Example 1.19\n')
printf('Refer to code for explanation\n\n')

N_1=300,N_2=375 //generator speeds

//E_g2=E_g1*(N_2/N_1)
//Using this new table OCC at N_2=375 is made
//Draw a line with slope R_sh=40 through origin
   which cuts this OCC at 248
//I_f_table=[0,2,3,4,5,6,7]
//Arm_vol_table
   =[9.375,115,165,202.5,228.75,237.5,265]

//part(i)
//at V=200 volts, I_f=3.9 from the graph
V=200
I_f=3.9
R_sh2=V/I_f,R_sh=40
printf('Additional resistance required is %.3f ohms
    ',R_sh2-R_sh)

//part(ii)
V_t=200
I_f=V_t/R_sh
E_g=228.75   //For this I_f from the table
R_a=0.4
I_a=(E_g-V_t)/R_a   //Because E_g=V_t + I_a*R_a
I_L=I_a-I_f
printf('\nLoad current supplied by the generator is
   %.3f A',I_L      )
```

Scilab code Exa 1.20 TO CALCULATE AMPERE TURNS AND SERIES TURNS TO BALANCE DEMAGNETISING COMPONENT OF A LAP CONNECTED GENERATOR

```
clc,clear
printf('Example 1.20\n\n')

I_a=750 //full load current
Pole=6
A=Pole //lap winding
I=I_a/A  //Full-load current per path
Z=900 //no of conductors
lambda=1.4  //leakage coefficient
theta_e=21 //lead angle in degrees electrical
theta_m=theta_e/(Pole/2)  //lead angle in degrees
   mechanical

amp_turns_PP_d=Z*I*theta_m/360  //demagnetising
   ampere turns per pole
amp_turns_PP_c=Z*I*(1/(2*Pole)-theta_m/360)  //cross
   -magnetising ampere turns per pole

balance_turns=amp_turns_PP_d*lambda/I_a //series
   turns required to balance demagnetising component
printf('(i) De-magnetising ampere-turns per pole is
    %.1f',amp_turns_PP_d)
printf('\n(ii) Cross-magnetising ampere-turns per
   pole is %.1f',amp_turns_PP_c)
printf('\n(iii)Turns required to balance
   demagnetising component is %.0f',floor(
   balance_turns))
```

Scilab code Exa 1.21 TO DESIGN A LAP WINDING

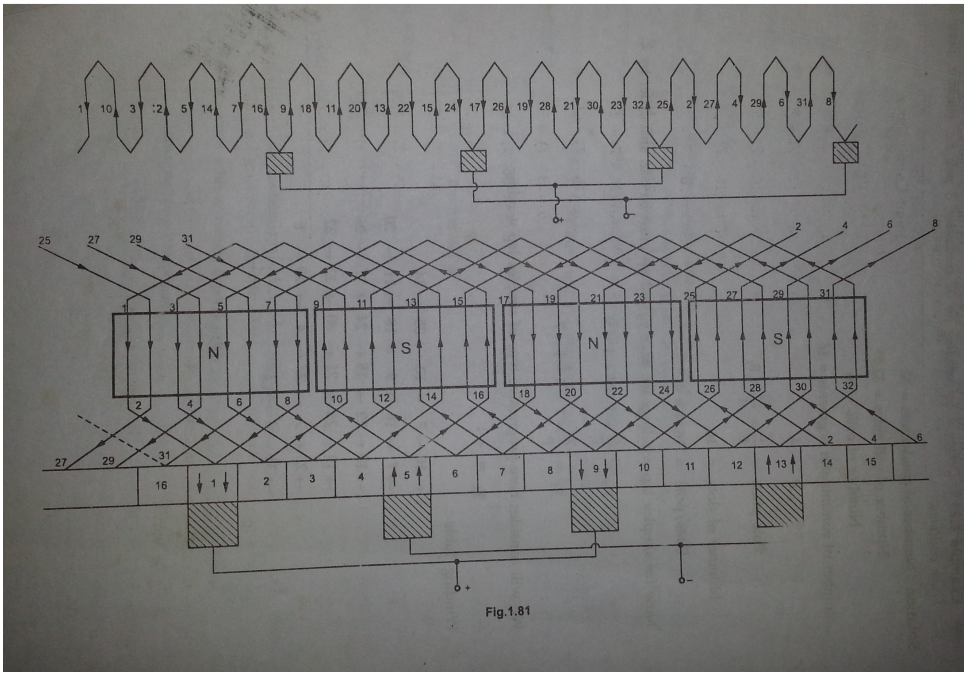


Figure 1.13: TO DESIGN A LAP WINDING

```
clc,clear
printf('Example 1.21\n\n')

Pole=4
Z=32  //no of conductors
coil_sides=Z
segments=16
pole_pitch=Z/Pole
slots=16
slots_per_pole=slots/Pole

//for Simplex lap winding
y_b=pole_pitch+1  //back pitch
y_f=pole_pitch-1  //front pitch

y_c=1  //Commutator pitch; Note that it is positive
   and it is progressive type of Simplex lap winding

printf('WINDING TABLE:\n\n    1<-    10->     3<-
   12->     5<-    14\n-> 7<-    16->     9<-    18->
   11<-    20\n->13<-    22->    15<-    24->    17<-
   26\n->19<-    28->    21<-    30->    23<-    32\n
   ->25<-     2->    27<-     4->    29<-     6\n->31<-
     8->1

   ')
printf('\nNote that <- indicates back connection
   with  y_back=%.0f and -> indicates front
   connection with y_front=%.0f\n',y_b,y_f)

printf('\nAnother form of winding table:')

printf('\n       BACK CONNECTIONS
                                        FRONT CONNECTIONS'
   )

printf('\n\n        1 to (1+9) =10
                        ->         10 to (10-7) =3')
```

```
26 printf('\n        3 to (3+9) =12
                        ->          12 to (12-7)= 5')
27 printf('\n        5 to (5+9) =14
                        ->          14 to (14-7)= 7')
28 printf('\n        7 to (7+9) =16
                        ->          16 to (16-7)= 9')
29 printf('\n        9 to (9+9) =18
                        ->          18 to (18-7)=11')
30 printf('\n       11 to (11+9)=20
                        ->          20 to (20-7)=13')
31 printf('\n       13 to (13+9)=22
                        ->          22 to (22-7)=15')
32 printf('\n       15 to (15+9)=24
                        ->          24 to (24-7)=17')
33 printf('\n       17 to (17+9)=26
                        ->          26 to (26-7)=19')
34 printf('\n       19 to (19+9)=28
                        ->          28 to (28-7)=21')
35 printf('\n       21 to (21+9)=30
                        ->          30 to (30-7)=23')
36 printf('\n       23 to (23+9)=32
                        ->          32 to (32-7)=25')
37 printf('\n       25 to (25+9)=34=(34-32)=2
                      ->            2 to (34-7)=27')
38 printf('\n       27 to (27+9)=36=(36-32)=4
                      ->            4 to (36-7)=29')
39 printf('\n       29 to (29+9)=38=(38-32)=6
                      ->            6 to (38-7)=31')
40 printf('\n       31 to (31+9)=40=(40-32)=4
                      ->            8 to (40-7)=33 -32= 1')
```

Scilab code Exa 1.22 TO DETERMINE CERTAIN QUANTITIES ASSOCIATED WITH SIMPLEX WAVE WOUND DC MACHINE

```
1 clc,clear
```

```
2  printf('Example 1.22\n\n')
3
4  Z=496 //no of conductors
5  P=4 //poles
6  slots=31
7  coilsides_per_slot=4
8  coilsides=slots*coilsides_per_slot
9  coils=coilsides/2
10 turns=Z/2
11 turns_per_coil=turns/coils
12
13 y_c=[(Z-2)/P (Z+2)/P] //commutator pitch
14 coils_active=(Z/(2*P))-1 // because y_c didnt turn
      out to be integer, 1 coil was made inactive/dummy
15 segments=coils_active  //no of commutative segments
16 Y_A=[ (segments+1)/(P/2) (segments-1)/(P/2)   ]
17 Y_A=Y_A(1)   //Y_A(2) is discarded because of
      progressive wave winding
18 y_f=29,y_b=33  //front and back pitch ; note that
      Y_A=(y_b+y_f)/2
19 resultant_pitch=2*Y_A  //because Y_A=(y_b+y_f)/2 and
       resultant pitch = y_b+ y_f
20
21 printf('\n(i)   Total number of coils = %.0f',coils)
22 printf('\n(ii)  Turns per coils = %.0f',
      turns_per_coil)
23 printf('\n(iii) Commutator pitch = %.0f',(y_c(1)+y_c
      (2))/2)
24 printf('\n(iv)  Back pitch= %.0f    front pitch= %.0
      f   total pitch= %.0f',y_b,y_f,resultant_pitch)
25 printf('\n(v)   No of commutator segments = %.0f',
      segments)
```

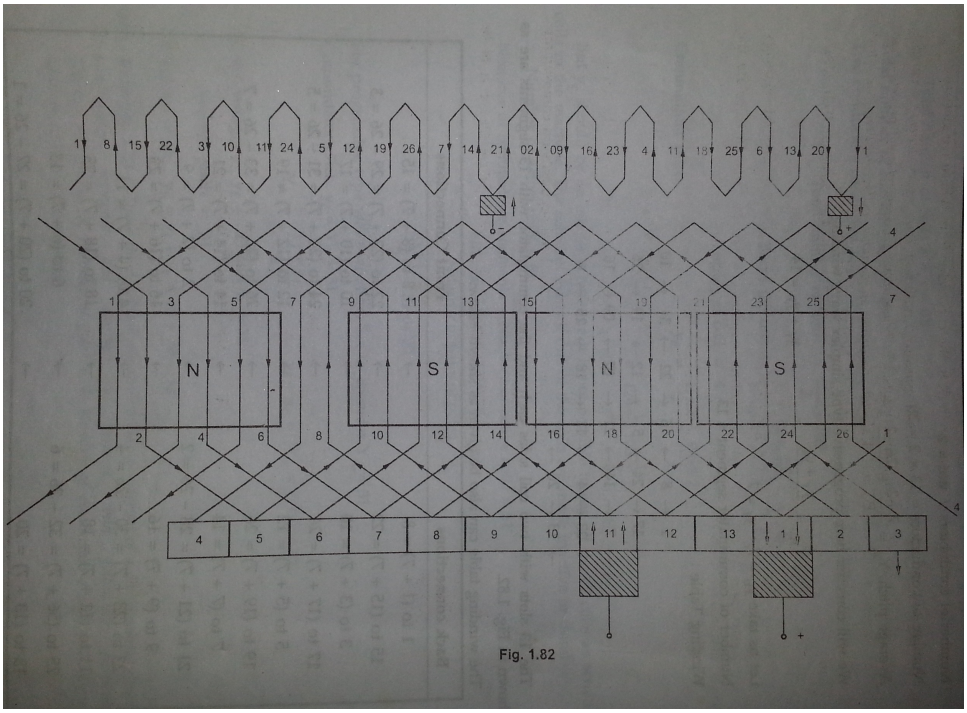


Figure 1.14: TO DRAW DEVELOPED ARMATURE WINDING DIAGRAM OF DC MACHINE

Scilab code Exa 1.23 TO DRAW DEVELOPED ARMATURE WINDING DIAGRAM OF DC MACHINE

```
1 clc,clear
2 printf('Example 1.23\n\n')
3
4 slots=13
5 Pole=4
6 conductors_per_slot=2
7 Z=conductors_per_slot*slots
8 Y_A=(Z+2)/Pole  //For progressive type wave winding
9 //Since Y_A=(y_b+y_f)/2, we let y_b=y_f
10 y_b=Y_A
11 y_f=y_b //because y_b=Y_A/2
12
13 segments=13 //commutator segments
14
15 printf('WINDING TABLE:\n    1<-     8->   15<-   22->
      3<-    10\n->17<-    24->    5<-   12->   19<-    26\n
      -> 7<-    14->   21<-    2->    9<-    16\n->23<-
      4->   11<-   18->   25<-     6\n->13<-    20->    1\n

      ')
16 printf('\nNote that <- indicates back connection
      with y_back=%.0f and -> indicates front
      connection with y_front=%.0f\n',y_b,y_f)
17
18 printf('\nAnother form of winding table:')
19
20 printf('\n       BACK CONNECTIONS
                                        FRONT CONNECTIONS'
      )
21
22 printf('\n\n         1 to (1+7)  = 8
                          ->          8 to (8+7) = 15')
23 printf('\n        15 to (15+7) =22
      ->         22 to (22+7)= 29 -26=3')
24 printf('\n         3 to (3+7)  =10
```

```
    ->        10 to (10+7)= 17')
printf('\n      17 to (17+7) =24
    ->        24 to (24+7)= 31 -26=5')
printf('\n       5 to (5+7)  =14
    ->        12 to (12+7)= 19')
printf('\n      19 to (19+7) =26
    ->        26 to (26+7)= 33 -26=7')
printf('\n       7 to (7+7)  =14
    ->        14 to (14+7)= 21')
printf('\n      21 to (21+7) =28 -26=2
    ->         2 to  (2+7)=  9')
printf('\n       9 to (9+7)  =16
    ->        16 to (16+7)= 23 ')
printf('\n      23 to (23+7) =30 -26=4
    ->         4 to  (4+7)= 11')
printf('\n       11to (11+7) =18
    ->        18 to (18+7)= 25')
printf('\n      25 to (25+7) =32 -26=6
    ->         6 to  (6+7)= 13')
printf('\n      13 to`(13+7) =20
    ->        20 to (20+7)= 27 -26=1')
```

Scilab code Exa 1.24 TO DETERMINE REACTIVE VOLTAGE IN CASE OF LINEAR AND SINUSOIDAL COMMUTATION

```
clc,clear
printf('Example 1.24\n\n')

P=4
I_L=150
N=1500 //commutator speed in rpm
n_s=N/60  //commutator speed in r.p.s
W_b=1.2 //Brush pitch
W_m=0//Pitch of mica insulation
L=0.05*10^-3  //inductance of armature coils in
```

```
    henry
A=P //A=P for lap wound
segments=64
v=n_s*segments //peripheral speed in segments per
    second

T_c=(W_b-W_m)/v  //Time of commutation
I=I_L/A  //current through each conductor

E=L*2*I/T_c  //Linear commutation
E2=1.11*L*2*I/T_c  //Sinusoidal commutation

printf('\nReactive voltage (linear commutation) is %
    .0f V',E)
printf('\nReactive voltage (sinusoidal commutation)
    is %.2f V',E2)
```

Scilab code Exa 1.25 TO CALCULATE ARMATURE CURRENT AND OUTPUT POWER

```
clc,clear
printf('Example 1.25\n\n')
printf('Note :answer obtained will not match with
    textbook answer because \nI_L=V_t/R_L\n
    =310.79/40=7.77 A\nwhile its taken as 8.045 A in
    textbook')

P=4  //Pole
Z=386 //no of wave connected conductors
A=2  //Wave winding
R_a=1,R_sh=100,R_L=40  //Armature ,shunt field and
    load resistance
phi=25*10^-3  //flux per pole in weber
```

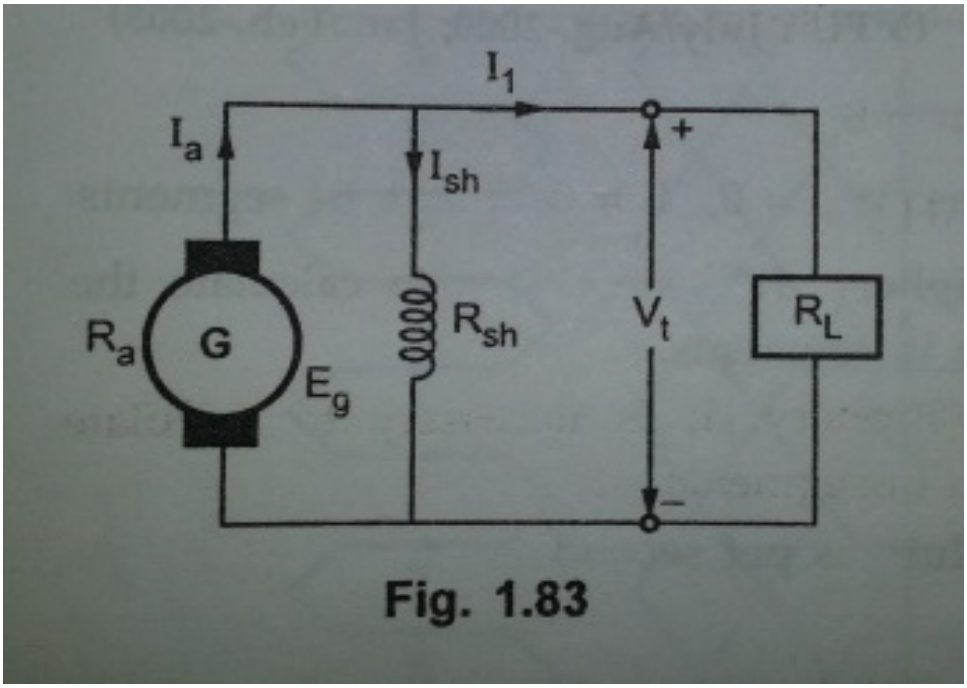


Figure 1.15: TO CALCULATE ARMATURE CURRENT AND OUTPUT POWER

```
10 N=1000  //speed in rpm
11
12 E_g=(phi*P*N*Z)/(60*A)  //generated emf
13
14 //Solving following equations for V_t
15 //E_g=V_t+I_a*R_a
16 //I_a=(I_L+I_sh)   I_L=V_t/R_L  I_sh=V_t/R_sj
17 //E_g=V_t(1 + (R_a/R_L) + (R_a/R_sh)
18 V_t=E_g/(1 + (R_a/R_L) + (R_a/R_sh))
19
20 I_L=V_t/R_L // load current
21 I_sh=V_t/R_sh // current through shunt field
      resistance
22
23 I_a=I_L+I_sh //armature current
24 printf('\n\narmature current is  is %.4f A',I_a)
25
26 output_power=V_t*I_L
27 printf('\noutput power is %.3f W',output_power)
```

Scilab code Exa 1.26 TO DETERMINE REACTIVE VOLTAGE FOR A DC MACHINE

```
1 clc,clear
2 printf('Example 1.26\n\n')
3
4 I=40  //current PER conductor
5 W_b=3,W_m=0//brush width and width of mica
      insulation
6 N=600 //commutator speed in rpm
7 n_s=N/60 //commutator speed in rps
8 L=0.15*10^-3 //self inductance in Henry
9 segments=50
10 v=n_s*segments //peripheral speed in segments per
      second
```

```
11 T_c=(W_b-W_m)/v //time of commutation
12
13 E=L*2*I/T_c  //Linear commutation
14 E2=1.11*L*2*I/T_c  //Sinusoidal commutation
15
16 printf('\nReactive voltage (linear commutation) is %
   .0f volts',E)
17 printf('\nReactive voltage (sinusoidal commutation)
   is %.2f volts',E2)
```

Scilab code Exa 1.27 TO CALCULATE CROSS AND DEMAGNETISING TURNS PER POLE

```
1 clc,clear
2 printf('Example 1.27\n\n')
3
4 V=400
5 P=6 //Poles
6 A=P  //For lap wound
7 output_power=250*10^3
8 R_sh=200 //shunt field circuit resistance
9 Z=720 //number of lap wound conductors
10
11 theta_m=2.5 //brush lead angle in degree mechanical
12 I_L=output_power/V
13 V_sh=V
14
15 I_sh=V_sh/R_sh //Current through shunt field circuit
    resistance
16 I_a=I_L+I_sh //armature currrent
17 I=I_a/P
18
19 //Part(i)
20 amp_turns_PP_d=Z*I*theta_m/360//demagnetising ampere
    turns per pole
```

```
21 //Part(ii)
22 amp_turns_PP_c=Z*I*(1/(2*P)-theta_m/360) //cross-
      magnetising ampere turns per pole
23
24 printf('(i) De-magnetising ampere-turns per pole is
      %.1f',amp_turns_PP_d)
25 printf('\n(ii) Cross-magnetising ampere-turns per
      pole is %.1f',amp_turns_PP_c)
```

Scilab code Exa 1.28 TO CALCULATE REACTIVE VOLTAGE IN CASE OF LINEAR COMMUTATION

```
1  clc,clear
2  printf('Example 1.28\n\n')
3
4  I_L=100
5  P=4 //Poles
6  A=P //for lap wound armature
7  W_b=1.4,W_m=0 //Brush width and width of mica
      insulation
8  N=1400//armature speed in r.p.m
9  segments=64 //no of commutator segments
10 L=0.05*10^-3 //inductance of armature coil in henry
11 n_s=N/60  //speed in r.p.s
12 v=n_s*segments //Segments per second
13 T_c=(W_b-W_m)/v //time of commutation
14 I=I_L/A   //Current through conductor
15 E=L*2*I/T_c  //Linear commutation
16
17 printf('\nReactive voltage considering linear
      commutation is %.2f volts',E)
```

Scilab code Exa 1.29 TO CALCULATE DEMAGNETISING AND CROSS MAGNETISING AMPERE TURNS PER POLE

```
clc,clear
printf('Example 1.29\n\n')

P=8 //Poles
A=2 //Wave wound armature
Z=480 //number of armature conductors
I_a=200
I=I_a/A

//Part(i)
theta_m=0 //Geometric nuetral axis
amp_turns_PP_d=Z*I*theta_m/360    //De-magnetising ampere-turns per pole
amp_turns_PP_c=Z*I*(1/(2*P)-theta_m/360) //Cross-magnetising ampere-turns per pole
printf('Part(i)\nDe-magnetising ampere-turns per pole is %.0f',amp_turns_PP_d)
printf('\nCross-magnetising ampere-turns per pole is %.0f\n\n',amp_turns_PP_c)

//Part(ii)
theta_e2=6 //angle shift of brushes in degrees electrical
theta_m2=theta_e2/(P/2) //angle shift of brushes in degrees mechanical
amp_turns_PP_d2=Z*I*theta_m2/360   //De-magnetising ampere-turns per pole
amp_turns_PP_c2=Z*I*(1/(2*P)-theta_m2/360) //Cross-magnetising ampere-turns per pole
printf('Part(ii)\nDe-magnetising ampere-turns per pole is %.0f',amp_turns_PP_d2)
printf('\nCross-magnetising ampere-turns per pole is %.0f',amp_turns_PP_c2)
```

Scilab code Exa 1.30 TO CALCULATE ARMATURE REACTION AMPERE TURNS AND DEMAGNETISING AND CROSSMAGENTISING AMPERE TURNS

```
1 clc,clear
2 printf('Example 1.30\n\n')
3 printf('The difference in answer occured because I
    is approximated to 16 in last 2 steps in book\n\n
    ')
4
5 P_input=7.46*10^3
6 V=230
7 Pole=8
8 Z=188 //number of armature consuctors
9 I_L=P_input/V
10 theta_m=7.5  //brush lead angle in degree mechanical
11
12 A=2 // assumed wave wound because of low-current and
     high voltage
13 I=I_L/A
14
15 //Part(i)
16 amp_turns_PP_d=Z*I*theta_m/360  //De-magnetising
    ampere-turns per pole
17 //Part(ii)
18 amp_turns_PP_c=Z*I*(1/(2*Pole)-theta_m/360) //Cross-
    magnetising ampere-turns per pole
19
20 printf('De-magnetising ampere-turns per pole is %.2f
    ',amp_turns_PP_d)
21 printf('\nCross-magnetising ampere-turns per pole is
    %.2f',amp_turns_PP_c)
```

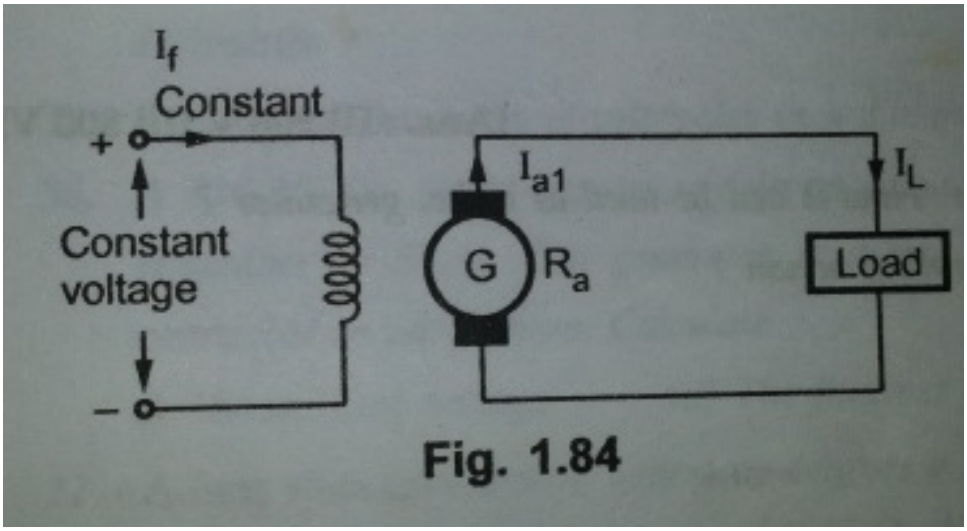


Figure 1.16: TO DETERMINE ALTERED CURRENT WHEN SPEED OF SEPERATELY EXCITED GENERATOR IS DROPPED

Scilab code Exa 1.31 TO DETERMINE ALTERED CURRENT WHEN SPEED OF SEPERATELY EXCITED GENERATOR IS DROPPED

```
1  clc,clear
2  printf('Example 1.31\n\n')
3
4  N_1=1200    //initial speed
5  I_L1=200    //initial load current
6  V_t1=125
7  N_2=1000   //altered speed
8  R_a=0.04   //armature resistance
9  V_brush=2   //brush drop
10
11 // Initial Load
12 I_a1=I_L1
```

```
13 E_g1=V_t1+I_a1*R_a+V_brush  //induced emf
14
15 E_g2=E_g1*(N_2/N_1)  //Because E_g proportional to N
       during constant flux
16 R_L= V_t1/I_L1 //Load resistance
17
18 //Solving for I_L2 as follows
19 //V_t2=R_L*I_L2   //I_a2=I_L2
20 //V_t2=E_g2-(I_L2*R_2 + V_brush)
21 I_L2=(E_g2-V_brush)/(R_L+R_a)  //new current
22
23 printf('Load current at new speed is %.4f A
                ',I_L2)
```

Chapter 2

DC Motors

Scilab code Exa 2.1 TO DETERMINE INDUCED EMF IN MOTOR

```
1 clc,clear
2 printf('Example 2.1\n\n')
3
4 V=220
5 I_a=30 //armature currnet
6 R_a=0.75 //Armature resistance
7
8 E_b=V - I_a*R_a    // Since V= E_b+ I_a*R_a
9 printf('Induced EMF or back EMF in the motor is %.1f
    V',E_b)
```

Scilab code Exa 2.2 TO CALCULATE BACK EMF AND MOTOR SPEED

```
1 clc,clear
2 printf('Example 2.2\n\n')
3
4 Pole=4
5 A=Pole //for lap winding
```

```
6 V=230
7 Z=250 //number of armature conductors
8 phi=30*10^-3 //flux per pole in weber
9 I_a=40,R_a=0.6 //Armature resistance
10
11 E_b=V - I_a*R_a    // Since V= E_b+ I_a*R_a
12 N=E_b * 60*A/(phi*Pole*Z)    //because E_b = phi*P*N*
     Z/(60*A)
13 printf('Back emf is %.0f V and running speed is %.0f
     rpm',E_b,N)
```

Scilab code Exa 2.3 TO DETERMINE GROSS TORQUE DEVELOPED BY MOTOR ARMATUTRE

```
1 clc,clear
2 printf('Example 2.3\n\n')
3
4 Pole=4
5 A=Pole //for lap winding
6 Z=480//number of armature conductors
7 phi=20*10^-3 //flux per pole in weber
8 I_a=50 //Armature current
9 T_a = 0.159*phi*I_a*Pole*Z/A  //Gross torque
     developed by armature
10 printf('Gross torque developed by armature is %.3f N
     -m',T_a)
```

Scilab code Exa 2.4 TO DETERMINE CERTAIN QUANTITIES RELATED TO LAP WOUND DC MOTOR

```
1 clc,clear
2 printf('Example 2.4\n\n')
3
```

```
4 Pole=4
5 A=Pole //for lap winding
6 V=230,R_a=0.8 //Armature resistance
7 N_0=1000 //no load speed in rpm
8 Z=540 //number of armature conductors
9 phi=25*10^-3 //flux per pole in weber
10 E_b0 = phi*Pole*N_0*Z/(60*A)  //induced emf
11
12 //part(i)
13 printf('(i)Induced e.m.f = %.0f V\n',E_b0)
14 //part(ii)
15 I_a0 = (V- E_b0)/R_a  //because V= E_b0+ I_a0*R_a
16 printf('(ii)Armature current = %.2f A\n',I_a0)
17 //part(iii)
18 stray_losses = E_b0*I_a0  //on no load ,power
     developed is fully power required to overcome
     strya losses
19 printf('(iii)Stray loss = %.2f W\n',stray_losses)
20 //part(iv)
21 T_f = E_b0*I_a0/(2*%pi*N_0/60)  //lost torque
22 printf('(iv)Lost torque = %.3f N-m\n',T_f)
```

Scilab code Exa 2.5 TO CALCULATE SPEED WHEN MOTOR DRAWS 60 A FROM SUPPLY

```
1 clc,clear
2 printf('Example 2.5\n\n')
3
4 Pole=4
5 Z=200 //No of armature conductors
6 A=2 //wave connected armature
7 V=250
8 phi=25*10^-3 //flux per pole in weber
```

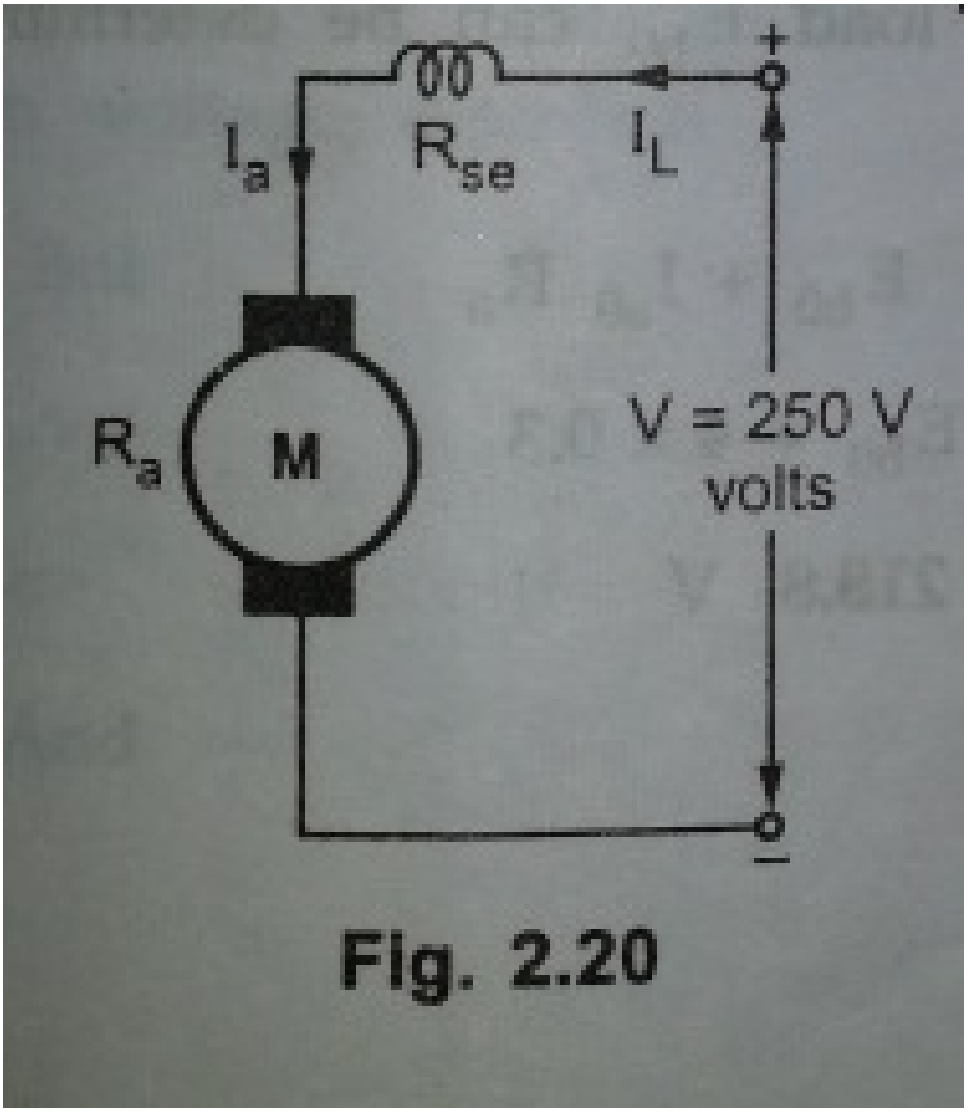


Figure 2.1: TO CALCULATE SPEED WHEN MOTOR DRAWS 60 A FROM SUPPLY

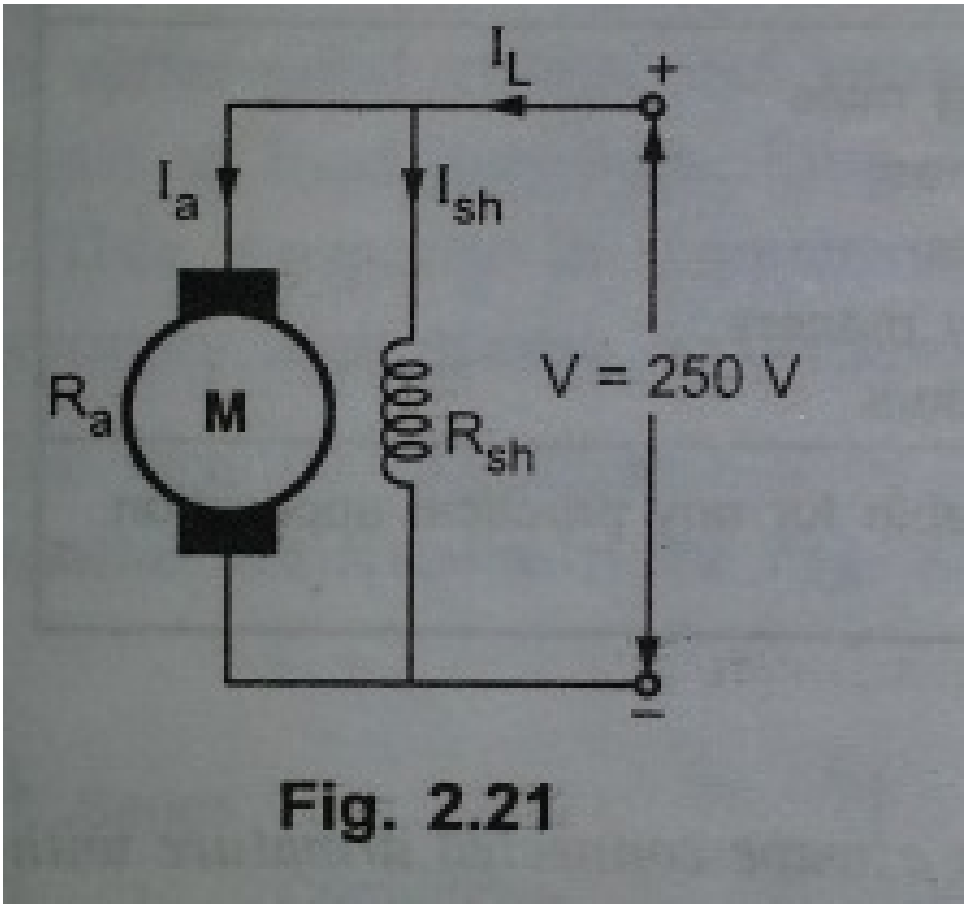


Figure 2.2: TO DETERMINE ARMATURE CURRENT AND BACK EMF

```
9  I_a =60, I_L =I_a  //armature current
10 R_a=0.15, R_se=0.2 //resistances of armature and
     series field winding
11 E_b= V - I_a*(R_a+R_se)  //induced emf
12 N=E_b * 60*A/(phi*Pole*Z)   //because E_b = phi*P*N*
     Z/(60*A)
13 printf('Required speed is %.0f r.p.m',N)
```

Scilab code Exa 2.6 TO DETERMINE ARMATURE CURRENT AND BACK EMF

```
clc,clear
printf('Example 2.6\n\n')

V=250
I_L =20 //load current
R_a=0.3, R_sh=200 //Armature and shunt field winding
I_sh=V/R_sh //shunt current
I_a=I_L-I_sh //armature current
E_b= V - I_a*R_a //emf generated
printf('Armature current is %.2f A\n',I_a)
printf('Back e.m.f is %.3f V',E_b)
```

Scilab code Exa 2.7 TO DETERMINE SPEED ON FULL LOAD

```
clc,clear
printf('Example 2.7\n\n')

V=220,R_a=0.3,R_sh=110 //resistance of armature and
   shunt field winding
//no load
N_0=1000 //no load speed in r.p.m
I_L0 =6 //line current on no load
I_sh= V/R_sh //no load shnt current
I_a0 = I_L0 - I_sh //no load armature current
E_b0= V - I_a0*R_a //no load induced emf

//full load
```

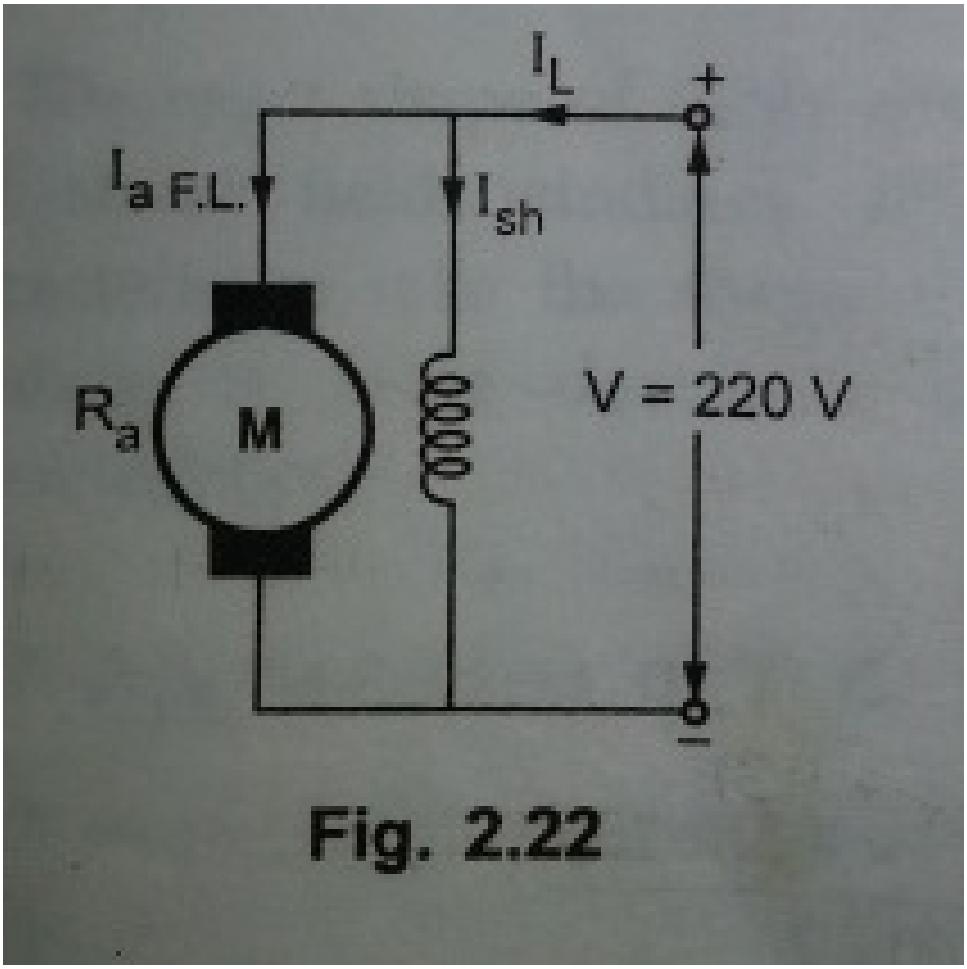


Figure 2.3: TO DETERMINE SPEED ON FULL LOAD

```
13 I_sh_FL= V/R_sh
14 I_L_FL=50 //line current at full load
15 I_a_FL= I_L_FL - I_sh_FL//full load armature current
16 E_b_FL= V - I_a_FL * R_a //full load induced emf
17 //using speed equation, as treating phi as constant
18 N_FL=N_0 * (E_b_FL/E_b0)
19 printf('Speed on full load is %.2f r.p.m',N_FL)
```

Scilab code Exa 2.8 TO DETERMINE SPEED OF MOTOR WITH ALTERED LOAD

```
1 clc,clear
2 printf('Example 2.8\n\n')
3
4 R_a=0.2, R_se =0.3 //Resistance of armature and
     series field winding
5 //following variables correspond to load 1
6 V=250
7 N_1=800
8 I_1=20, I_a1=I_1,I_se1= I_a1
9 E_b1= V - I_a1*(R_a+R_se)
10 //following variables correspond to load 2
11 I_2=50, I_a2=I_2
12 E_b2= V - I_a2*(R_a+R_se)
13
14 //from speed equation it can be derived that,
15 N_2 = N_1 * (E_b2/E_b1) * (I_a1/I_a2)
16 printf('Speed on motor on no load  is %.0f r.p.m',
     N_2)
```

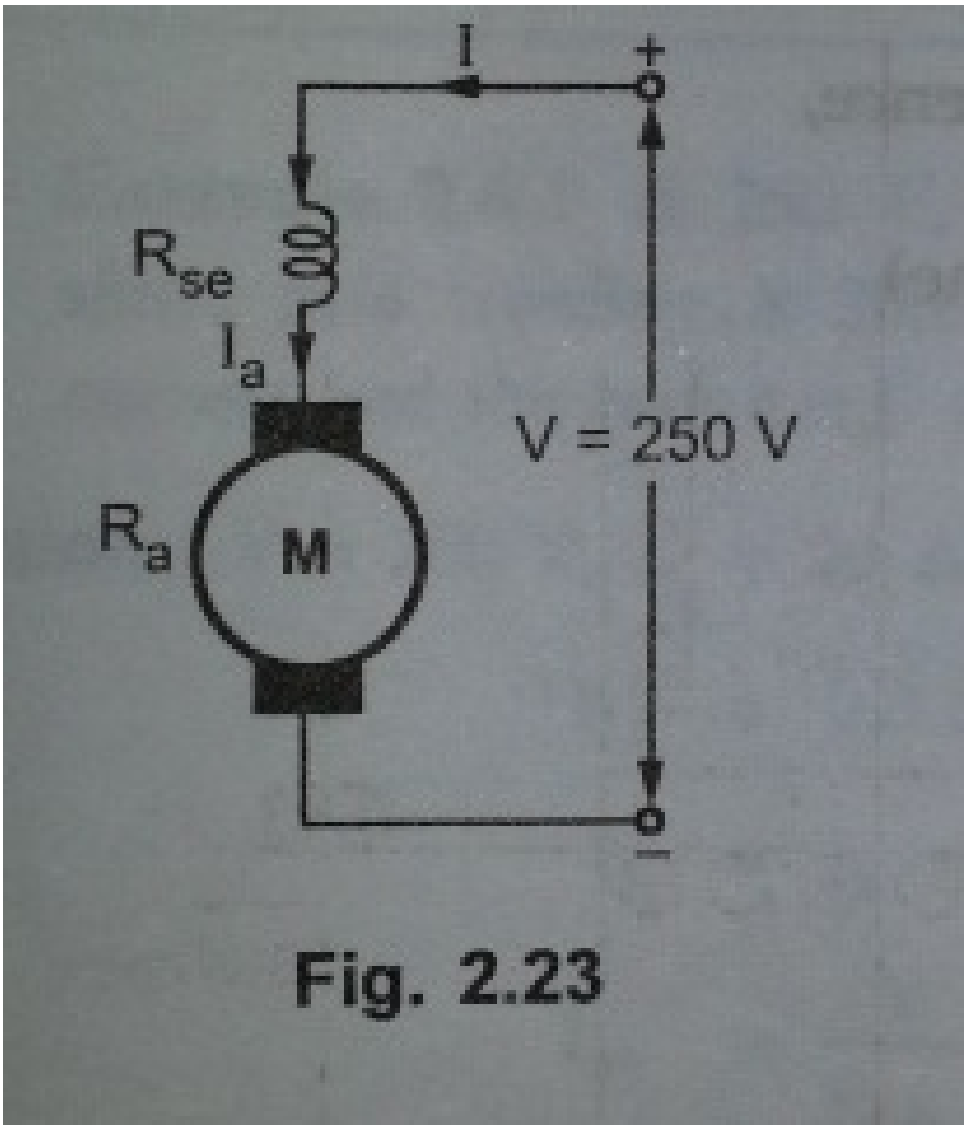


Figure 2.4: TO DETERMINE SPEED OF MOTOR WITH ALTERED LOAD

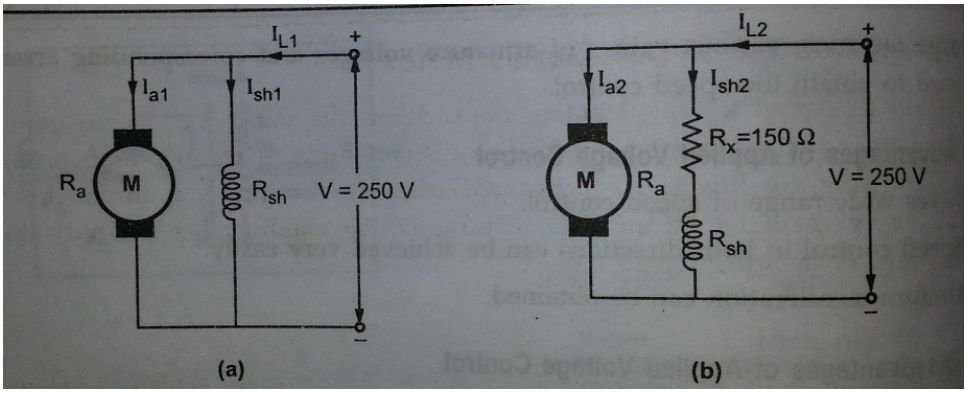


Figure 2.5: TO DETERMINE ARMATURE CURRENT WHEN A RESISTANCE IS ADDED IN SERIES TO FIELD WINDING

Scilab code Exa 2.9 TO DETERMINE ARMATURE CURRENT WHEN A RESISTANCE IS ADDED IN SERIES TO FIELD WINDING

```
1  clc,clear
2  printf('Example 2.9\n\n')
3
4  V=250
5  R_a=0.3,R_sh=200  //resistance of armature and shunt
       field winding
6  R_x=150 //additional resistance added in series to
       field winding
7  I_L1=22
8  I_sh1=V/R_sh //initial shunt current before adding
       150 ohms resistance
9  I_a1 = I_L1 - I_sh1 //initial armature current
       before adding 150 ohms resistance
10 N_1=1500 //initial speed before adding 150 ohms
       resistance
11 //T (prop.) phi*I_a (prop.) I_sh*I_a and T_1=T_2 and
       simplifying further
```

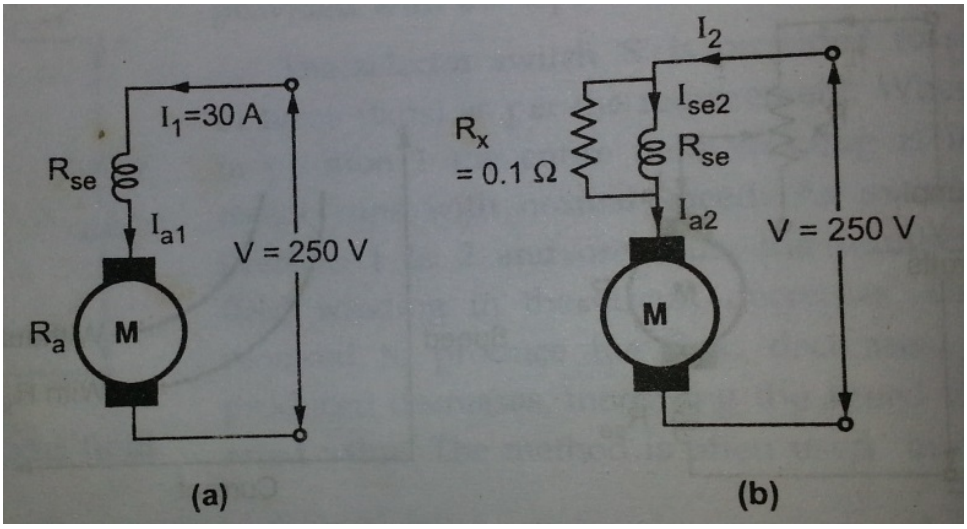


Figure 2.6: TO DETERMINE MOTOR SPEED WHEN FIELD WINDING GETS SHUNTED BY A RESISTANCE

```
12 I_sh2=V/(R_sh + R_x) //new shunt current
13 I_a2= I_sh1*I_a1/I_sh2  //New armature current
14
15 E_b1=V - I_a1*R_a    //induced emf before adding 150
      ohms resistance
16 E_b2=V - I_a2*R_a    //new emf
17
18 N_2 = N_1 * (E_b2/E_b1) * (I_sh1/I_sh2) //new speed
19 printf('New armature current and speed are %.4f A
      and %.2f r.p.m respectively',I_a2,N_2)
```

Scilab code Exa 2.10 TO DETERMINE MOTOR SPEED WHEN FIELD WINDING GETS SHUNTED BY A RESISTANCE

```
1 clc,clear
2 printf('Example 2.10\n\n')
```

```
3
4 V=250
5 R_a=0.15, R_se=0.1, R_x=0.1 //Resitance of armature
    , series field winding and extra resistance
6 N_1 = 800 //initial speed before load torque is
    increased
7 I_1= 30 , I_a1=I_1 , I_se1 = I_1  //initial currents
8
9 T_2_by_T_1 = 1 + (50/100)  //50 percent increase as
    mentioned in question
10 I_se2_by_I_a2 = R_x/(R_x + R_se)  //from the figure
11
12 //T (prop.) phi*I_a (prop.) I_sh*I_a and T_1=T_2 and
     simplifying ,solving further
13 I_a2=sqrt(I_a1*I_se1*T_2_by_T_1/I_se2_by_I_a2) //new
     armature current
14 I_se2 = I_se2_by_I_a2 *I_a2 //new series field
    current
15
16 E_b1 = V - I_a1*R_a - I_se1*R_se //indiced emf
    initially
17 E_b2 = V - I_a2*R_a - I_se2*R_se //new induced emf
18 N_2 = N_1 * (E_b2/E_b1) * (I_se1/I_se2) //required
    speed
19 printf('The required running speed of motor is %.3f
    r.p.m',N_2)
```

Scilab code Exa 2.11 TO DETERMINE EXTRA RESISTANCE THAT WILL REDUCE THE SPEED

```
1 clc,clear
2 printf('Example 2.11\n\n')
3
```

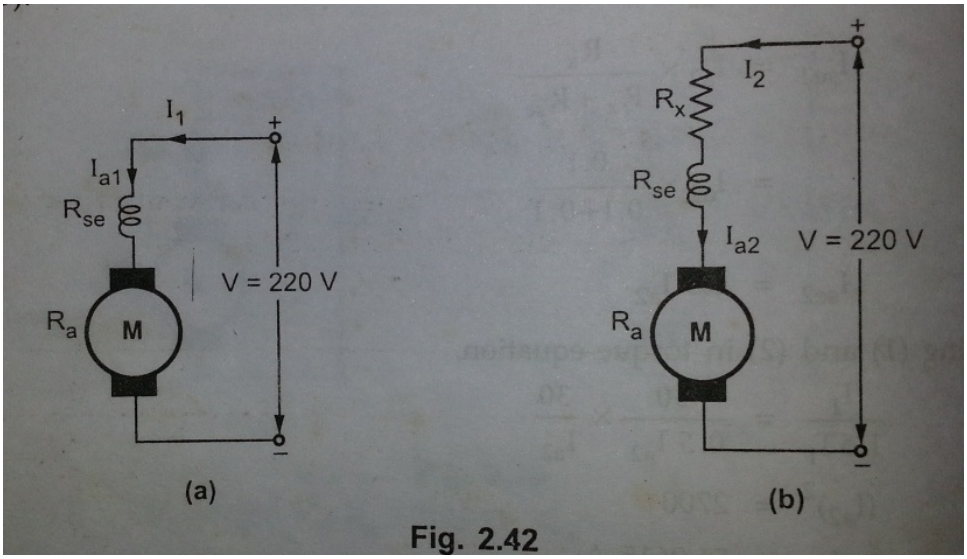


Figure 2.7: TO DETERMINE EXTRA RESISTANCE THAT WILL REDUCE THE SPEED

```
4  V=220
5  I_1=50, I_a1=I_1  //Currents before adding extra
       resistance
6  T_2_by_T_1 =0.5
7  R_t=0.15 //R_e + R_se =0.15
8
9  I_a2 =I_a1 * sqrt(T_2_by_T_1)  //Because T (prop.)
       I_a^2
10 E_b1=V-I_a1*(R_t) //induced emf before adding extra
       resistance
11 N_1=500,N_2=300 //speeds before and adding extra
       resistance
12 //N (prop.) E_b/phi (prop.) E_b/I_a
13 E_b2=E_b1 *(I_a2/I_a1)*(N_2/N_1) //induced emf after
       adding resistance
14 R_x= (V-E_b2)/I_a2 -R_t //because E_b2=V - I_a2*(R_a
       + R_se + R_x)
15 printf('Desired extrea resistance= %.4f ohms ',R_x)
```

Scilab code Exa 2.12 TO DETERMINE CERTAIN QUANTITIES RELATED TO PERMANENT MAGNET DC MOTOR

```
1 clc,clear
2 printf('Example 2.12\n\n')
3
4 R_a= 1, I_a=1.2 , V=50
5 //part(i)
6 E_b = V - I_a*R_a
7 rot_loss_NL =E_b*I_a //no load rotational loss
8 printf('(i)No load rotational losses = %.2f W',
    rot_loss_NL)
9
10 //part(ii)
11 omega_2000=2*%pi*2000/60 //angular velocity when
    speed of motor =2000 rpm
12 K_m=E_b/omega_2000   //to determine K_m
13 V=48
14 omega_1800=2*%pi*1800/60 //angular velocity when
    speed of motor =1800 rpm
15 E_b=K_m*omega_1800
16 I_a = (V-E_b)/R_a  //armature current
17 P_dev = E_b*I_a//power developed
18 motor_output =  P_dev - rot_loss_NL
19 printf('\n(ii)Motor output = %.2f W',motor_output)
20
21 //part(iii)
22 E_b=0 //when motor stalls
23 V_stall=20 //voltage during stalling
24 I_a=V_stall/R_a //armature current during stalling
25 T_stall = K_m*I_a //stalling torque
26 printf('\n(iii)Stalling torque = %.2f N-m',T_stall)
27 printf('\n\npart(ii) answer is slightly different
    due to inaccurate calculation of Power developed'
```

```
)
```

Scilab code Exa 2.13 TO DETERMINE SPEED ON HALF LOAD CONDITION

```
clc,clear
printf('Example 2.13\n\n')

V=120
R_a=0.2 , R_sh=60 //armature and field resistance
I_L1=40, N_1=1800
I_sh= V/R_sh

I_a1=I_L1 - I_sh
E_b1 = V -I_a1*R_a  //Induced emf at half load
T2_by_T1 =1/2
I_a2=I_a1*(T2_by_T1)   //T (prop.) I_a
E_b2=V- I_a2*R_a//induced emf at half load
N_2 = N_1 *(E_b2/E_b1)  //N (prop.) E_b as phi is
   constant
printf('Speed on half load condition is %.2f r.p.m',
   N_2)
```

Scilab code Exa 2.14 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR

```
clc,clear
printf('Example 2.14\n\n')

R_a=0.08, E_b1=242 , V=250
//part(i)
```

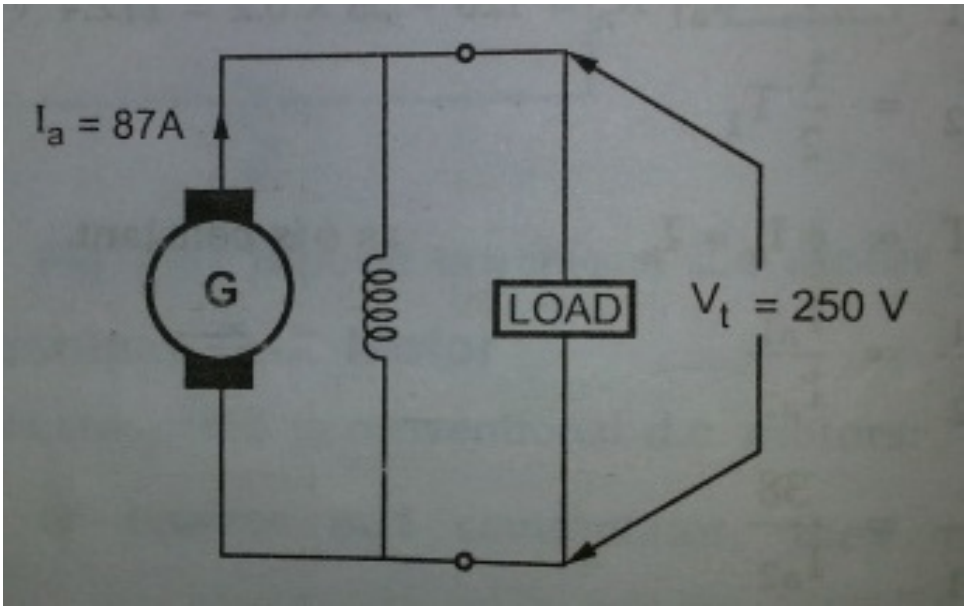


Figure 2.8: TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR

```
6  I_a1= (V-E_b1)/R_a
7  printf('(i)Armature current = %.0f A',I_a1)
8
9  //part(ii)
10 N=0
11 E_b=0 //because N=0
12 I_a_start=V/R_a
13 printf('\n(ii)Starting armature current = %.0f A',
      I_a_start)
14
15 //part(iii)
16 I_a2=120
17 E_b2=V-I_a2*R_a
18 printf('\n(iii)Back emf if armature current is
      changed to 120 A=   %.1f V',E_b2)
19
20 //part(iv)
21 I_a=87,N_m=1500
```

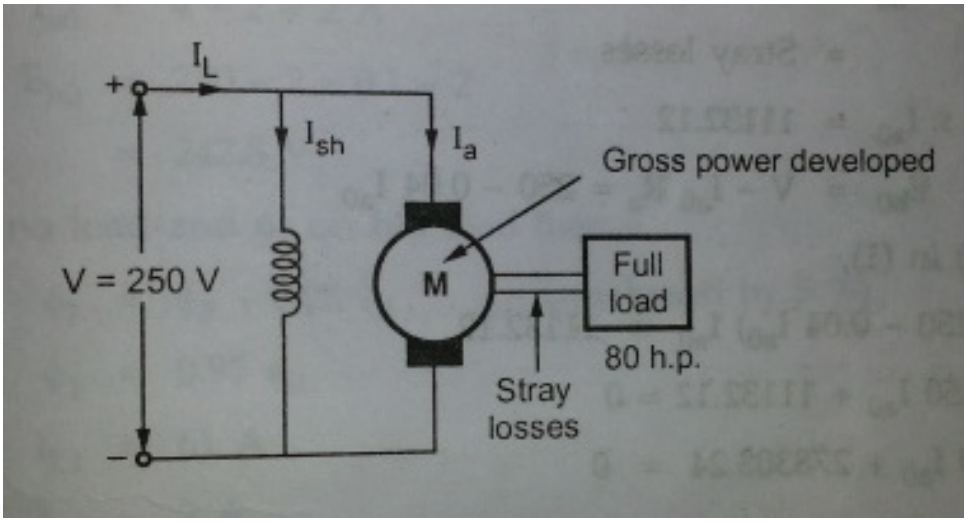


Figure 2.9: TO DETERMINE MECHANICAL POWER AND NOLOAD SPEED AND CURRENT

```
22 E_g=V + I_a*R_a //induced emf
23 N_g=N_m*(E_g/E_b1)//as E (prop.) N
24 printf('\n(iv)Generator speed to deliver 87 A at 250
       V = %.1f rpm',N_g)
```

Scilab code Exa 2.15 TO DETERMINE MECHANICAL POWER AND NOLOAD SPEED AND CURRENT

```
1 clc,clear
2 printf('Example 2.15\n\n')
3
4 shaft_output = 80*746 //coverted to watts
5 eta= 80/100 //efficiency
6 V=250
7 N_1=1200
```

```
R_a=0.04,R_sh = 250 //armature and shunt field
    resistance
power_input = shaft_output/eta
I_L= power_input /V
I_sh= V / R_sh
I_a = I_L - I_sh
E_b1 = V - I_a*R_a

gross_mechanical_power= E_b1*I_a //electrical
    equivalent of mechanical power developed
stray_losses =  gross_mechanical_power -
    shaft_output
printf('Mechanical power developed on full load = %
    .3f kW\n',gross_mechanical_power/1000)

//on no load shaft_output=0 and entire gross power
    is used to overcome stray losses
Eb0_Ia0= stray_losses
//E_b0 = V - I_a0*R_a  ... solving for I_0
p=[R_a -V Eb0_Ia0]
roots(p)
I_a0=ans(2) //first root is ignored since its too
    large
I_L0 =I_sh+I_a0 //current drawn from supply
E_b0 = V - I_a0*R_a

//From speed equation N (prop.) E_b
N_0 = N_1*(E_b0/E_b1)
printf('No load speed and current are %.4f rpm and %
    .2f A respectively',N_0,I_L0)
```

Scilab code Exa 2.16 TO DETERMINE FULL LOAD SPEED

```
clc,clear
printf('Example 2.16\n\n')
```

```
3
4 V=250, P=4
5 R_a=0.1 , R_sh =124 //armature and shunt field
    resistance
6 I_L0=4,N_0=1200
7 I_L_1=61
8 I_sh=V/R_sh
9 I_a0=I_L0-I_sh
10 V_brush= 2 //voltage loss due to brush
11 E_b0= V - I_a0*R_a- V_brush
12
13 I_a1=I_L_1 - I_sh
14 E_b1=V - I_a1*R_a -V_brush
15
16 phi1_by_phi0=1-(5/100)   //weakened by 5 %
17 N_1 = N_0 *(E_b1/E_b0) /phi1_by_phi0
18
19 printf('Full load speed is %.3f r.p.m',N_1)
```

Scilab code Exa 2.17 TO DETERMINE CERTAIN QUANTITIES RELATED TO DC SHUNT MOTOR

```
1 clc,clear
2 printf('Example 2.17\n\n')
3
4 V=250
5 R_a=0.15 , R_sh=167.67 //armature and shunt field
    resistance
6 N_0=1280 //speed at no load
7
8 //full load
9 I_L1 = 67  //current drawn on full load
10 I_sh = V / R_sh  //as shunt motor
11 I_a1= I_L1- I_sh
12 E_b1= V - I_a1*R_a
```

```
13
14 //on no load
15 I_L0=6.5
16 I_a0 = I_L0 - I_sh
17 E_b0 = V - I_a0*R_a
18
19 //part(i) USING SPEED EQUATION
20 //N (prop.) E_b/phi (prop.)E_b   //as phi is
      constant
21 N_1 = N_0 * (E_b1 / E_b0)
22 printf('(i)Full load speed = %.3f r.p.m\n',N_1)
23
24 //part(ii)
25 speed_regulation = 100* ((N_0-N_1)/N_1)
26 //N_1 is full load speed and N_0=No load speed
27 printf('(ii)Speed regulation = %.2f percent \n',
      speed_regulation )
28
29 //part(iii)
30 shaft_output_FL = E_b1*I_a1 - E_b0*I_a0  //full load
       power developed - stray losses
31 hp_rating = shaft_output_FL /746
32 printf('(iii)HP rating of machine = %.2f h.p\n',
      hp_rating)
33
34 //part(iv)
35 power_input= V*I_L1
36 eta= 100*(shaft_output_FL/power_input)  //full load
      efficiency
37 printf('(iv)Full load efficiency = %.2f percent',eta
      )
```

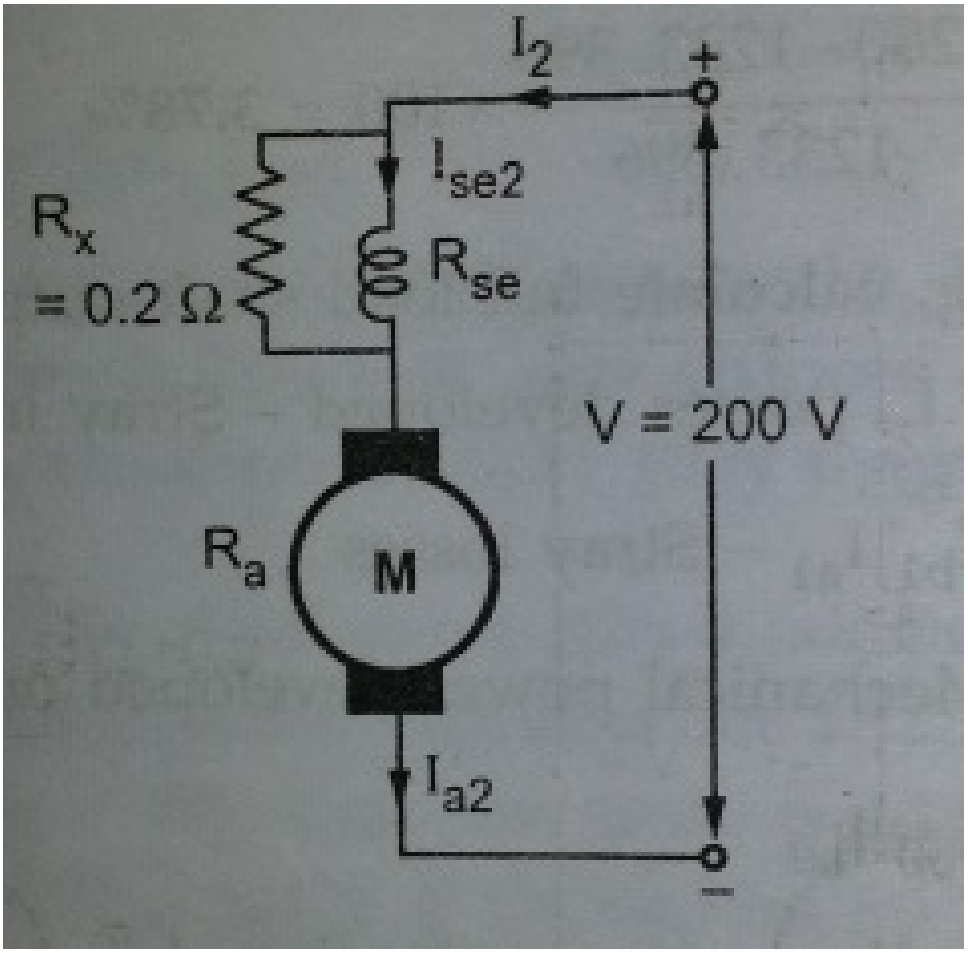


Figure 2.10: TO DETERMINE SPEED IF FIELD WINDING IS SHUNTED BY ADDITIONAL RESISTANCE

Scilab code Exa 2.18 TO DETERMINE SPEED IF FIELD WINDING IS SHUNTED BY ADDITIONAL RESISTANCE

```
1 clc,clear
2 printf('Example 2.18\n\n')
3
4 V=200
5 R_a=0.5, R_se=0.2, R_x=0.2 //armature and series
     field resistance; extra resistance
6 I_a1=20, I_1=I_a1 , I_se1=I_a1
7 I_a2=20, I_2=I_a2
8 I_se2= I_2 *(R_x/(R_se+R_x))
9
10 E_b1 = V -I_a1*R_a - I_a1*R_se
11 E_b2 = V -I_a2*R_a - I_se2*R_se
12
13 phi2_by_phi1=70/100
14 N_1=1000
15 N_2=N_1*(E_b2/E_b1)  /phi2_by_phi1          //N (prop
     .) E_b/phi
16 printf('Required speed is %.2f r.p.m',N_2)
```

Scilab code Exa 2.19 TO DETERMINE SPEED IF FIELD GROUPS ARE ARRANGED IN PARALLEL

```
1 clc,clear
2 printf('Example 2.19\n\n')
3
4 V=110
5 P=4
6 R_a = 0.1,R=0.01  //A resistance of 0.01 ohms
7 R_se=R+R
8
```

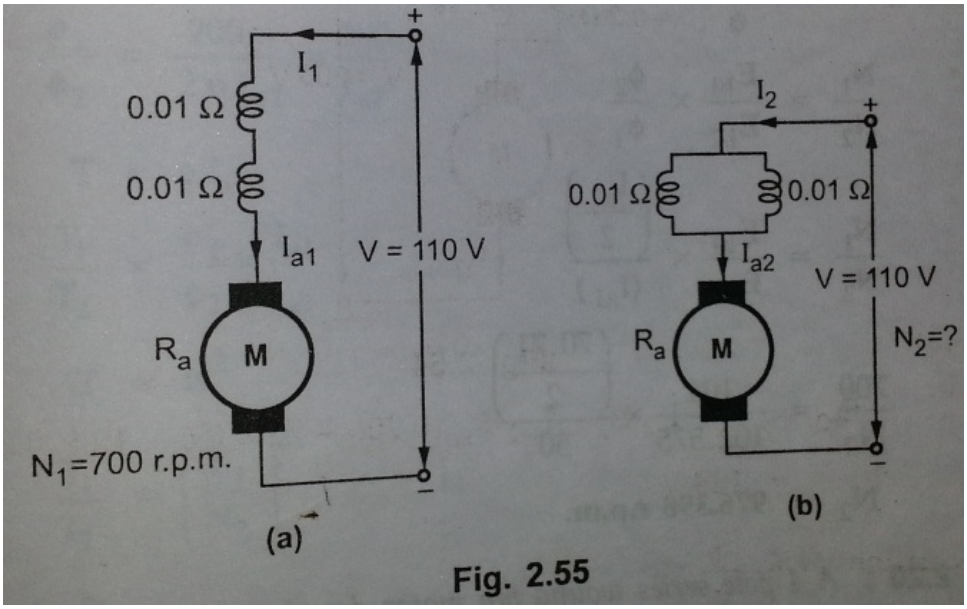


Figure 2.11: TO DETERMINE SPEED IF FIELD GROUPS ARE ARRANGED IN PARALLEL

```
9  //case(i)
10 I_1=50, I_a1=I_1
11 N_1=700
12 E_b1 = V -I_a1*(R_a + R_se)
13
14 //T (prop.) phi*I_a   from torque equation
                                        (1)
15
16 //phi_1 (prop.) I_a1

    (2)
17 //case(ii) when I_a2 gets divided to half
18 //phi_2 (prop.) I_a2/2

    (3)
19
20 //combining (1)(2)(3) and T1=T2
21 I_a2 = sqrt(2*I_a1^2)
```

```
22 R_se_eqvt=(R*R)/(R+R)    //Equavalent of parallel
     combination
23 E_b2 = V - I_a2*R_a - I_a2* R_se_eqvt
24
25 //Using speed equation N (prop.) E_b / phi  and
     using (2) and (3)
26 N_2 = N_1 *( E_b2/E_b1) *(I_a1/(I_a2/2))
27 printf('Speed after reconnection = %.3f r.p.m\n\n',
     N_2)
```

Scilab code Exa 2.20 TO DETERMINE NEW SPEED AND ARMATURE CURRENT AFTER RECONNECTION

```
1 clc,clear
2 printf('Example 2.20\n\n')
3
4 P=4, I_a1= 50, N_1=2000, V=230
5
6 //phi_1 is proportioanl to total ampere-turns
     produced by field coils
7 //phi_1 (prop.) I_a1*P*n (prop.) 200*n
                                              (1)
8
9 //After reconnection, phi_2 proportional to ampere
     turns divided as follows
10 //phi_2 (prop.) [I_a2/2*2*n + I_a2/2*2*n]  (prop.)
     2*n*I_a2                    (2)
11
12 // Dividing (1) and (2) , (phi_1/phi_2)=100 / I_a2
                                  (3)
13
14 //T (prop.) phi*I_a AND T (prop.) N^2
                                              (4),(5)
```

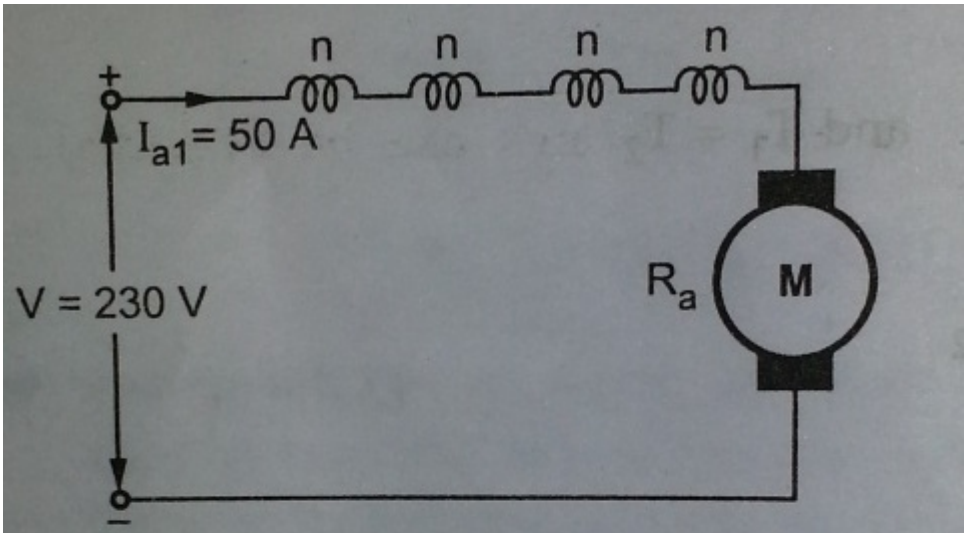


Fig. 2.56

∴

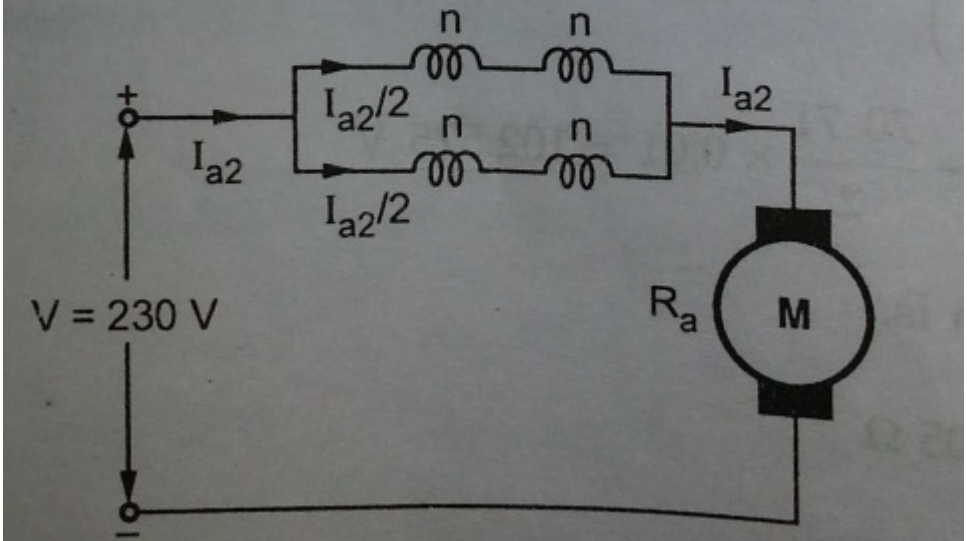


Fig. 2.57

Figure 2.12: TO DETERMINE NEW SPEED AND ARMATURE CURRENT AFTER RECONNECTION

```
15 //therefore N^2 (prop.) phi*I_a
      (6)
16
17 //N (prop.) E_b/phi (prop.) 1/phi    ..
18 //Because drops across windings can be neglected,
      E_b1=E_b2
19 //therefore N (prop.) 1/phi
      (7)
20
21 //Using (7) and (6) phi^3 (prop.) 1/I_a
                                                (8)
22
23 //combining (3) and (8)
24 I_a2 =  (50*100^3)^(1/4)     //new armature current
25 printf('New armature current= %.3f A\n',I_a2)
26 //combining (6) and (7)    , N^3 (prop.) I_a1
27 N_2=N_1 *(I_a2/I_a1)^(1/3)
28 printf('New motor speed =%.3f r.p.m',N_2)
```

Scilab code Exa 2.21 TO PROVE THAT PROPORTIONALITY CONSTANT IS SAME IN CASE OF BACK EMF and ARMATURE SPEED AND TORQUE AND ARMATURE CURRENT

```
1 clc,clear
2 printf('Example 2.21\n\n')
3
4 //  K_1= E_b/N   = (phi*P*Z)/(60*A)
5
6 //P_m = T * omega
7 //E_b*I_a = T *(2*%pi*N/60)
8 //T= I_a  (E_b*60 / 2*%pi*N )
9 //Use E_b= phi*P*N*Z/(2*%pi*A)
10 //T / I_a = phi*P*Z / (2*%pi*A)  =K_dasah
```

```
11
12 printf('The constant of proportionality in both the
      cases is  K=K_dash = phi*P*Z /(2*3.142*A)')
```

Scilab code Exa 2.22 TO CALCULATE EXTRA RESISTANCE TO REDUCE THE SPEED

```
1  clc,clear
2  printf('Example 2.22\n\n')
3
4  V=200,I_a1=30
5  R_t=1.5  //R_a + R_se
6  E_b1= V - I_a1*R_t
7  N2_by_N1=(60/100)
8
9  //T (prop.) I_a^2  and T (prop.) N_3.... therefore
      I_a^2 (prop.) N^3
10 I_a2  = I_a1*sqrt(N2_by_N1^3)
11
12 //N (prop.) E_b/I_a
13 N2_by_N1
14 E_b2 = E_b1 *(I_a2/I_a1)*N2_by_N1
15 R_x= (V- E_b2)/I_a2 - R_t        //because E_b2= V -
       I_a2*(R_x+R_t)
16 printf('Additional resistance to be added in series
      with motor circuit = %.3f ohms',R_x)
```

Scilab code Exa 2.23 TO DETERMINE ADDITIONAL RESISTANCE IN FIELD CIRCUIT TO RAISE THE SPEED

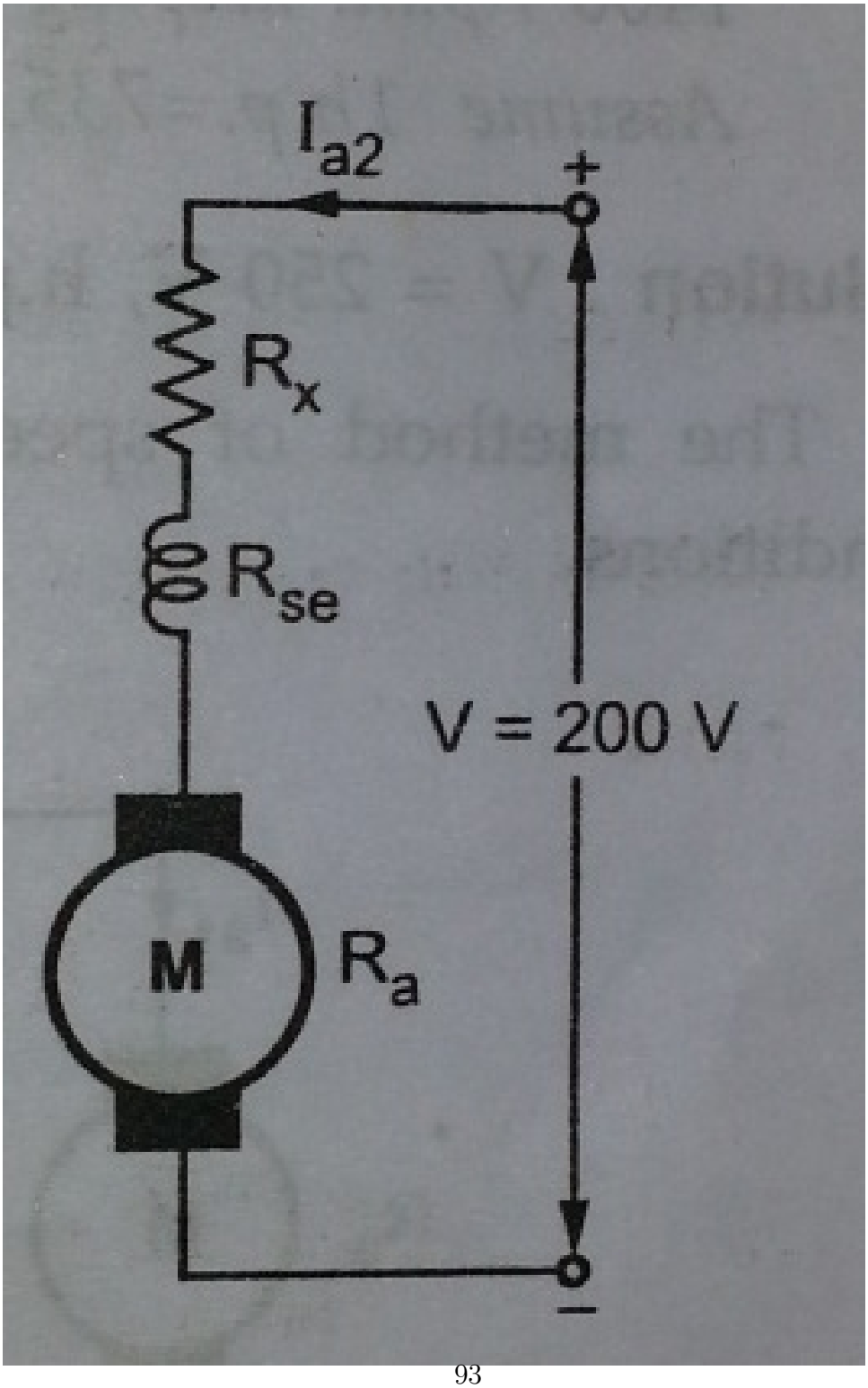


Figure 2.13: TO CALCULATE EXTRA RESISTANCE TO REDUCE THE SPEED

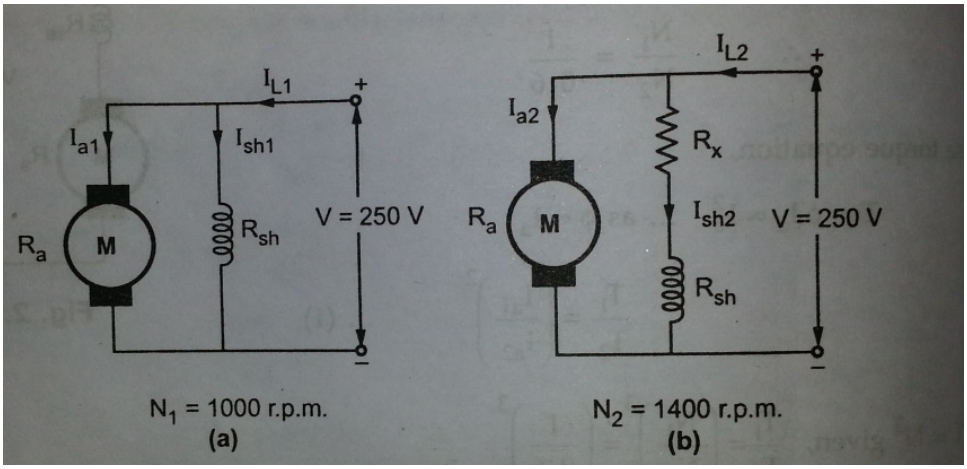


Figure 2.14: TO DETERMINE ADDITIONAL RESISTANCE IN FIELD CIRCUIT TO RAISE THE SPEED

```
clc,clear
printf('Example 2.23\n\n')

V=250,
R_a=0.4 , R_sh=100 //armature and shunt field
    resistance
I_sh1=V / R_sh
P_out_FL = 10 * 735.5
eta=85/100 //efficiency
P_in= P_out_FL/eta
I_L1= P_in /V
I_a1= I_L1 - I_sh1

// T (prop.) phi*I_a (prop.) I_sh*I_a    because phi
    (prop.) I_sh
//Bu torque is constant..
Ia2_Ish2= I_a1*I_sh1
E_b1= V - I_a1*R_a

//N (prop.) E_b/I_sh
//put E_b2= V - I_a2*R_a  and solving further for
```

```
    I_sh2 we get, I_sh2^2 - 1.8824 I_sh2 +0.2417=0
20 p=[1 -1.8824 0.2417]
21 roots(p)
22 I_sh2=ans(1)
23 //root 1 was considered because its always easier to
       attain root(1) because less resistacne is
    needeed
24 //R_x in series with field
25 R_x = (V/I_sh2) -R_sh    //because I_sh2 = V/(R_sh +
    R_x)
26 printf('Extra Resistance to be added  = %.2f ohms',
    R_x)
```

Scilab code Exa 2.24 TO DETERMINE SUPPLY VOLTAGE REQUIRED TO RAISE FAN SPEED

```
1  clc,clear
2  printf('Example 2.24\n\n')
3
4  R_t=1 //R_t = R_se + R_a
5  V_1= 230
6  N_1=300,N_2=375
7  I_1=15, I_a1=I_1
8
9  //T (prop.) I_a^2  and T (prop.) N_2....therefore
       I_a^2 (prop.) N^2
10 I_a2=I_a1 *(N_2/N_1)
11 E_b1 = V_1 - I_a1*(R_t)
12
13 //N (prop.) E_b/I_a
14 E_b2= E_b1*(I_a2/I_a1)*(N_2/N_1)
15 V_2=E_b2 + I_a2* (R_t)  //because E_b2 = V_2 - I_a2
    *(R_a+R_se)
16 printf('Voltage supply needed = %.4f V',V_2)
```

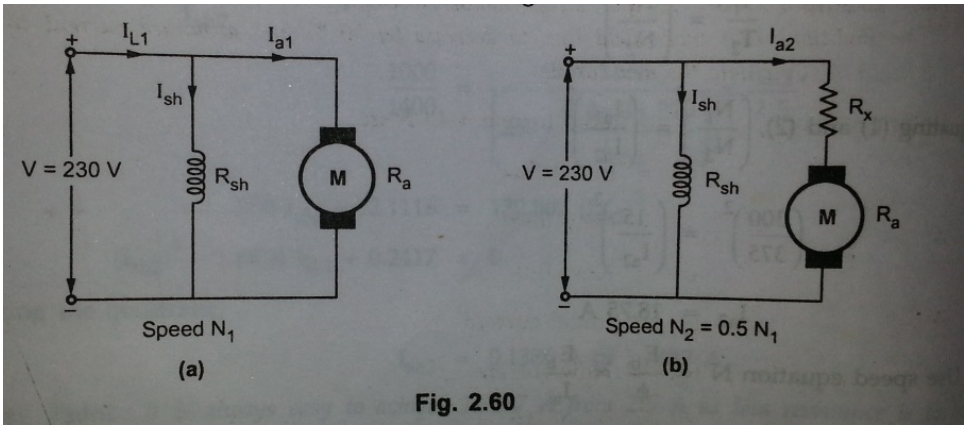


Figure 2.15: TO CALCULATE RESISTANCE TO BE CONNECTED IN SERIES WITH ARMATURE TO HALVE THE SPEED

Scilab code Exa 2.25 TO CALCULATE RESISTANCE TO BE CONNECTED IN SERIES WITH ARMATURE TO HALVE THE SPEED

```
1  clc,clear
2  printf('Example 2.25\n\n')
3
4  I_L1=30,V=230
5  R_sh=230,R_a=1
6  I_sh= V / R_sh
7  I_a1= I_L1 - I_sh
8  E_b1 = V - I_a1*R_a
9
10 //T (prop.) phi*I_a (prop.) I_a    as phi is constant
11 //and torque is constant
12 I_a2 = I_a1
13 N2_by_N1= 1/2
14 //N (prop.) E_b/phi (prop.) E_b
```

```
15 E_b2= E_b1 *(N2_by_N1)
16 R_x= (V- E_b2)/I_a2 - R_a      //Because E_b2 = V -
      I_a2*(R_a + R_x)
17 printf('Resistance to be inserted in series = %.4f
      ohms ',R_x)
```

Scilab code Exa 2.26 TO CALCULATE TORQUE ALTERED DUE TO CHANGES IN FIELD FLUX AND ARMATURE CURRENT

```
1 clc,clear
2 printf('Example 2.26\n\n')
3
4 T_1=40  //initial torque
5 //phi_1 is initial flux
6 //phi_2 is new flux
7 //T_2 is new torque
8 //I_a1 is initial current
9 //I_a2 is new current
10 phi2_by_phi1 = 1- (30/100)  //decrease by 30 percent
11 Ia2_by_Ia1=1+(15/100)       //increase by 15 percent
12
13 //T (prop.) phi*I_a
14 T_2=T_1*(phi2_by_phi1)*(Ia2_by_Ia1)
15 printf('New torque is %.1f N-m',T_2)
```

Scilab code Exa 2.27 TO CALCULATE EXTRA RESISTANCE IN SERIES WITH ARMATURE TO REDUCE SPEED AT FULL LOAD

```
1 clc,clear
2 printf('Example 2.27\n\n')
3
```

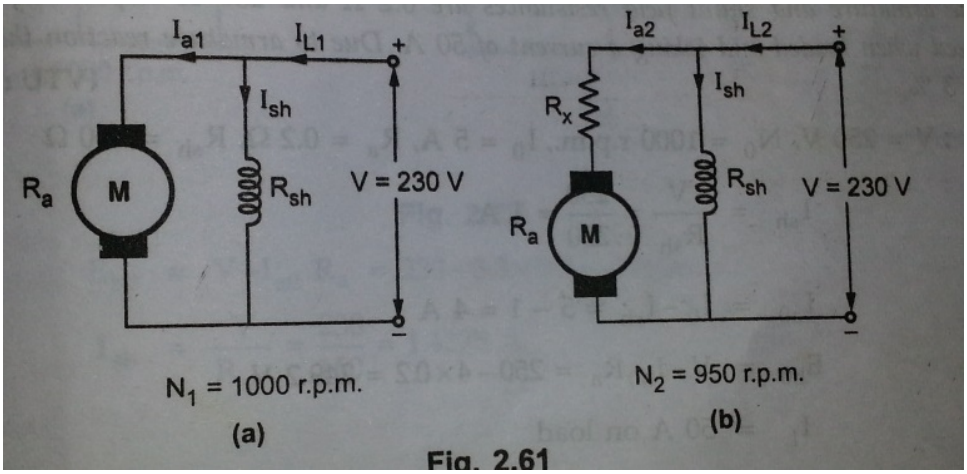


Figure 2.16: TO CALCULATE EXTRA RESISTANCE IN SERIES WITH ARMATURE TO REDUCE SPEED AT FULL LOAD

```
4  V=230
5  N_1=1000,N_2=950
6  R_a=0.5, R_sh=230 //armature and shunt field
       resistance
7  I_L1=10
8
9  I_sh = V/R_sh
10 I_a1 = I_L1 - I_sh
11
12 //T (prop.) phi*I_a  (prop.) I_a with phi constant
       and T is constant due to full-load
13 I_a2=I_a1
14
15 E_b1 = V - I_a1*R_a
16 E_b2=E_b1*(N_2/N_1) //N (prop.) E_b /phi (prop.) E_b
        as phi is constant
17
18 R_x = (V-E_b2)/I_a2  -R_a
19 printf('Resistance to be inserted in series with
       armature = %.4f ohms',R_x)
```

Scilab code Exa 2.28 TO DETERMINE SPEED WHEN DC SHUNT MOTOR GETS LOADED

```
clc,clear
printf('Example 2.28\n\n')

V=250,N_0=1000,I_0=5
R_a=0.2,R_sh=250 //armature and shunt field
   resistance
I_L=50 //on no load
I_sh=V / R_sh
I_a0 = I_0 - I_sh
I_a  = I_L - I_sh
E_b0 = V- I_a0*R_a
E_b1 = V- I_a *R_a

phi1_by_phi0 =1-(3/100) //weakens by 3 percent
//N (prop.) E_b/phi
N_1 = N_0 *(E_b1/E_b0) /phi1_by_phi0
printf('Speed when loaded and drawing 50A current is
    %.3f r.p.m',N_1)
```

Scilab code Exa 2.29 TO DETERMINE SPEED AND TORQUE DEVELOPED AT FULL LOAD WHEN NO LOAD FLUX WEAKENS

```
clc,clear
printf('Example 2.29\n\n')

V=230,I_a0=3.3
```

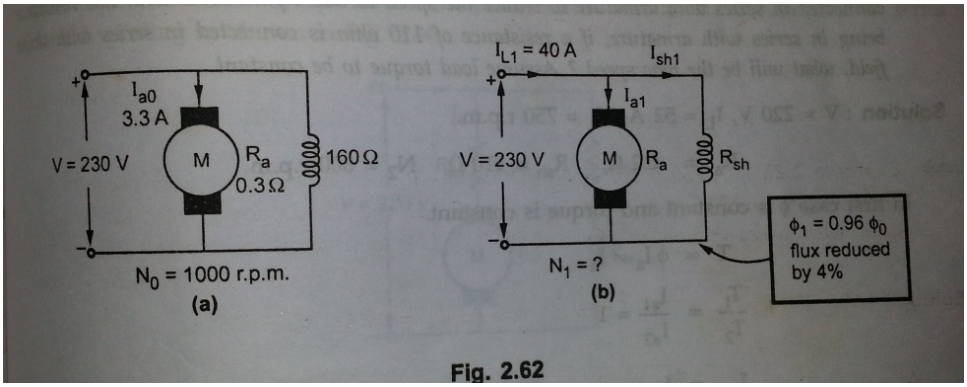
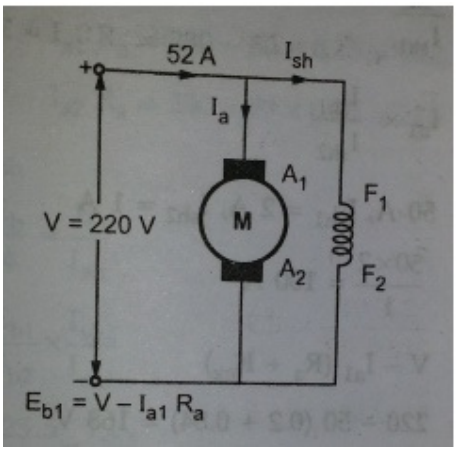


Figure 2.17: TO DETERMINE SPEED AND TORQUE DEVELOPED AT FULL LOAD WHEN NO LOAD FLUX WEAKENS

```
5  R_a=0.3,R_sh=160 //armature and shunt field
       resistance
6  I_L1=40,N_0=1000
7  E_b0 = V - I_a0*R_a
8  I_sh=V/ R_sh
9  I_a1 = I_L1 - I_sh
10 E_b1 = V - I_a1*R_a
11 phi1_by_phi0= 1- (4/100) //weakening by 4 percent
12
13 N_1 = N_0 *(E_b1/E_b0)/(phi1_by_phi0)  //because N (
       prop.) E_b/phi
14 printf('Full load speed is %.4f rpm\n',N_1)
15 T_0 = E_b0*I_a0/(2*%pi*N_0/60)
16 T_1 = T_0*(I_a1/I_a0)*phi1_by_phi0   // because T (
       prop.) phi*I_a
17 printf('Full load developed torque is %.4f N-m',T_1)
```

Scilab code Exa 2.30 TO FIND THE SPEED WHEN ADDITIONAL RESISTANCES GET CONNECTED WITH SHUNT FIELD AND ARMA-



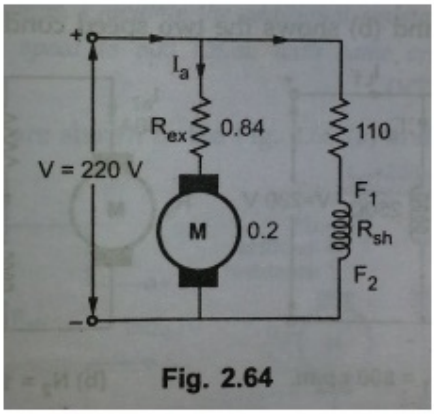


Figure 2.18: TO FIND THE SPEED WHEN ADDITIONAL RESISTANCES GET CONNECTED WITH SHUNT FIELD AND ARMATURE

TURE

```
clc,clear
printf('Example 2.30\n\n')

V=220
I_L=52
N_1=750, N_2=600
R_a=0.2, R_sh = 110 //armature and shunt field
    resistance

I_sh=V/ R_sh
I_a1= I_L - I_sh
I_a2=I_a1//T (prop.) I_a and T is constant
E_b1 = V - I_a1*R_a

//N (prop.) E_b/phi (prop.) E_b
E_b2 = E_b1*(N_2/N_1)
R_x  = (V- E_b2)/I_a2 -R_a  //Because E_b2 = V -
    I_a2*(R_a+R_x)
printf('Resistance to be connected in series = %.2f
    ohms\n',R_x)
```

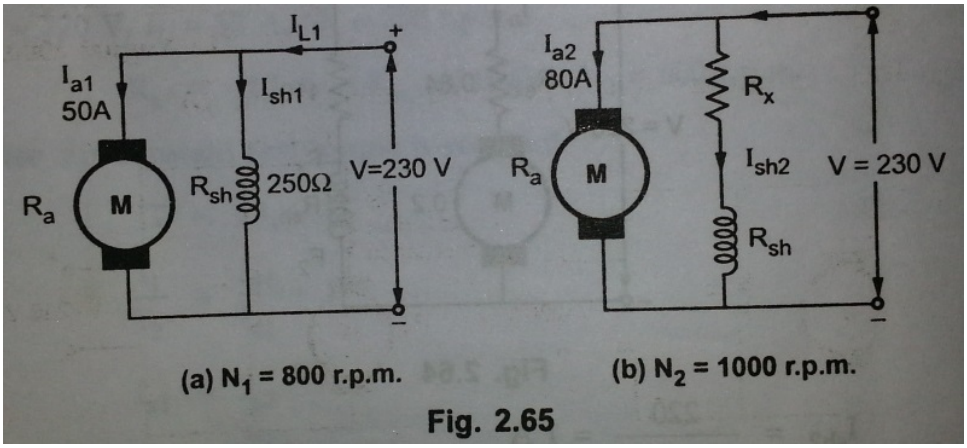


Figure 2.19: TO DETERMINE EXTRA RESISTANCE WITH FIELD CURRENT TO INCREASE SPEED OF DC SHUNT MOTOR

```
18
19 //After R_x gets connected in series with armature
     and 110 ohms in series with field winding
20 N_1=600
21 I_sh2=V /(R_sh+110)
22 I_a1=50,I_sh1=2,I_sh2=1
23 //T (prop.) I_a*I_sh and T doesn't vary
24 I_a2 = I_a1*(I_sh1/I_sh2)
25 E_b1 = V - I_a1*(R_a+R_x)
26 E_b2 = V - I_a2*(R_a+R_x)
27 N_2 = N_1*(E_b2/E_b1)*(I_sh1/I_sh2) //Because N (
     prop.) E_b/I_sh
28 printf('New speed= %.3f rpm',N_2)
```

Scilab code Exa 2.31 TO DETERMINE EXTRA RESISTANCE WITH FIELD CURRENT TO INCREASE SPEED OF DC SHUNT MOTOR

```
1 clc,clear
```

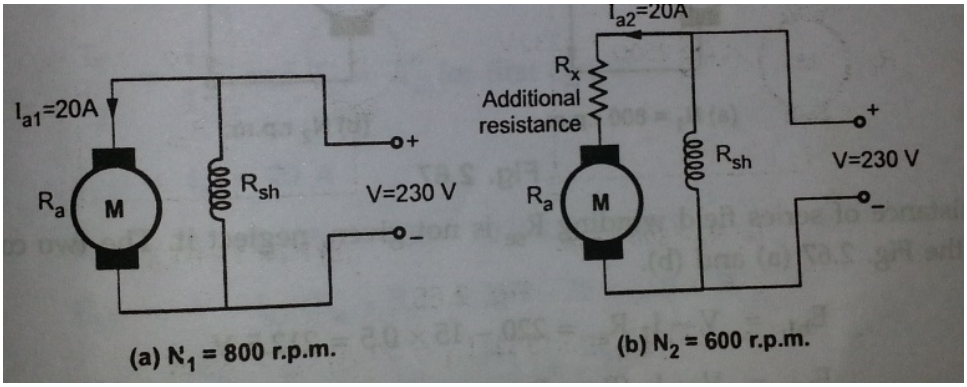


Figure 2.20: TO FIND EXTRA RESISTANCE TO BE ADDED IN SERIES WITH ARMATURE TO REDUCE ITS SPEED WITH SAME ARMATURE CURRENT

```
2  printf('Example 2.31\n\n')
3
4  V=230
5  R_a=0.15,R_sh=250 //armature and shunt field
       resistance
6  I_a1=50, I_a2= 80
7  N_1=800, N_2=1000
8  I_sh1= V / R_sh
9
10 E_b1 = V - I_a1*R_a
11 E_b2 = V - I_a2*R_a
12
13 I_sh2=I_sh1*(E_b2/E_b1)*(N_1/N_2) //Because N (prop
       .) E_b/ I_sh
14 R_x= (V/I_sh2 ) - R_sh     //because I_sh2 = V /(R_x+
       R_sh)
15 printf('Resistance to be added is \n\nR_x=%.0f ohms'
       ,R_x)
```

Scilab code Exa 2.32 TO FIND EXTRA RESISTANCE TO BE ADDED IN SERIES WITH ARMATURE TO REDUCE ITS SPEED WITH SAME ARMATURE CURRENT

```
1 clc,clear
2 printf('Example 2.32\n\n')
3
4 V=230,R_a=0.5
5 N_1=800,N_2=600
6 I_a2 =20 , I_a1=I_a2
7 E_b1 = V - I_a1*R_a
8
9 //N (prop.) E_b/phi (prop.) E_b          as phi is
    constant
10 E_b2=E_b1 *(N_2/N_1)
11 //additional resistance required
12 R_x = (V -E_b2)/I_a2  - R_a    //because E_b2 = V -
    I_a2*(R_a+R_x)
13 printf('Additional resistance required = %.2f ohms '
    ,R_x)
```

Scilab code Exa 2.33 TO DETERMINE THE SPEED WHEN ADDITIONAL RESISTANCE GETS CONNECTED AND DRAWING SAME CURRENT

```
1 clc,clear
2 printf('Example 2.33\n\n')
3
4 V=220
5 R_a=0.5,R_x=5 //armature resistacne and extra
    resistance
6 I_1=15, I_se1=I_1, I_se2=I_se1 , I_2=I_se2
7 N_1=800
8
```

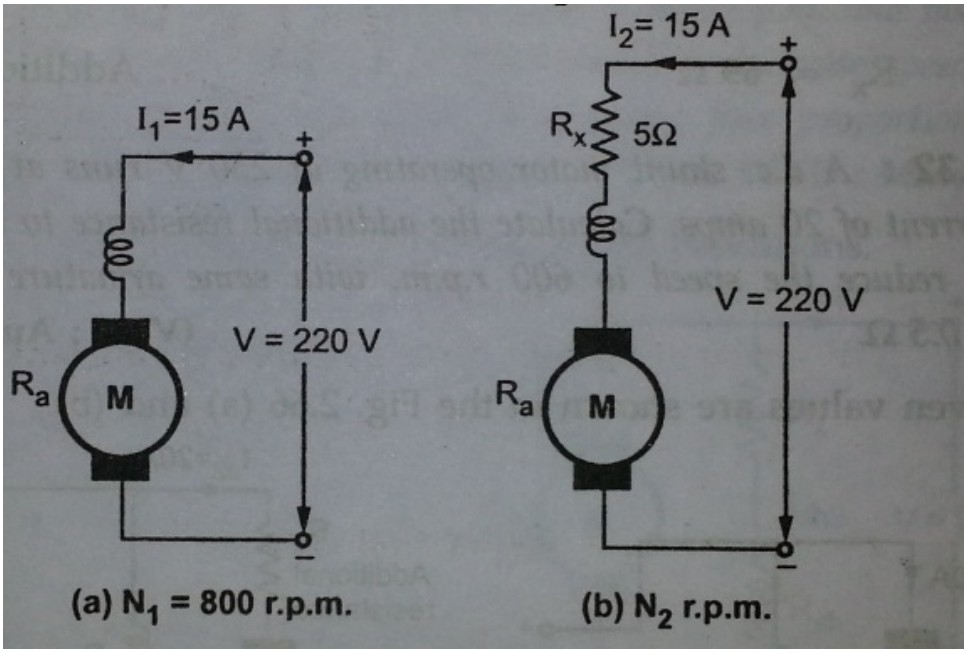


Figure 2.21: TO DETERMINE THE SPEED WHEN ADDITIONAL RESISTANCE GETS CONNECTED AND DRAWING SAME CURRENT

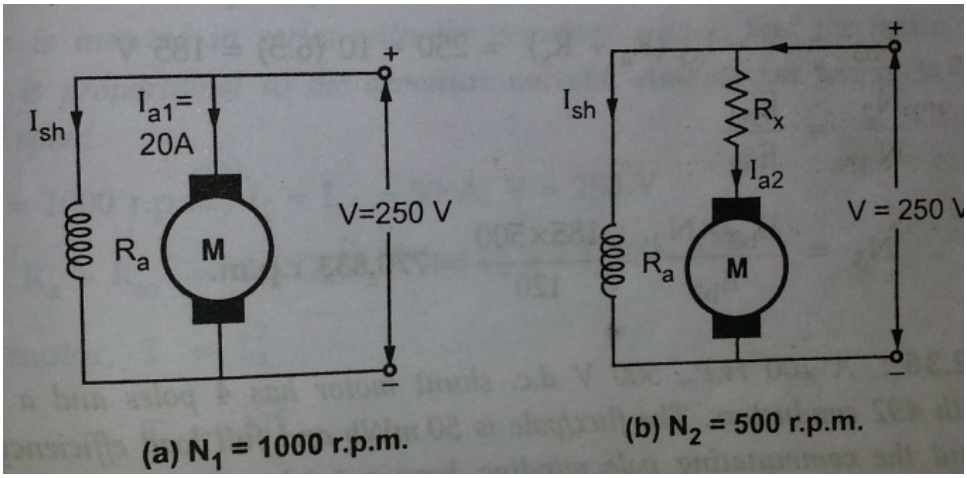


Figure 2.22: TO DETERMINE ADDITIONAL RESISTANCE IN SERIES WITH ARMATURE TO REDUCE THE SPEED AND ALTERED SPEED WHEN TORQUE GETS HALVED

```
9  E_b1 = V - I_1*R_a
10 E_b2 = V - I_2*(R_a+R_x)
11
12 N_2= N_1*(E_b2/E_b1)*(I_se1/I_se2) //because N (prop
       .) E_b/I_se
13 printf('New speed of rotor = %.3f r.p.m',N_2)
```

Scilab code Exa 2.34 TO DETERMINE ADDITIONAL RESISTANCE IN SERIES WITH ARMATURE TO REDUCE THE SPEED AND ALTERED SPEED WHEN TORQUE GETS HALVED

```
1 clc,clear
2 printf('Example 2.34\n\n')
3
4 V=250, I_a1=20, R_a=0.5
5 N_1=1000, N_2=500
```

```
6
7 //T (prop.) I_a and T_1=T_2
8 I_a2=I_a1
9 E_b1 = V - I_a1*R_a
10
11 //N (prop.) E_b
12 E_b2= E_b1 *(N_2/N_1)
13 R_x= (V-E_b2)/I_a2 - R_a    //because E_b2 = V - I_a2
     *(R_a+R_x)
14 printf('Additional resistance = %.0f ohms',R_x)
15 T3_by_T2=0.5    //torque is halved
16 I_a3= I_a2 *(T3_by_T2)    //new armature current
17 E_b3 = V - I_a3*(R_x + R_a)
18 N_3=E_b3*N_2 / E_b2 //N (prop.) E_b
19 printf('\nNew speed = %.3f rpm',N_3)
```

Scilab code Exa 2.35 TO CALCULATE SPEED AND USEFUL TORQUE ON FULL LOAD

```
1 clc,clear
2 printf('Example 2.35\n\n')
3
4 P_out= 100*735.5
5 V=500
6 P=4
7 A=2// due to wave winding
8 Z=492 //no of conductors
9 phi=50*10^-3 //flux per pole
10 eta=92/100 //efficiency
11 P_in= P_out/eta
12 R_a=0.1 , R_sh=250 //amature and shunt field
     resistance
13
14 I_L=P_in/V
15 I_sh = V/ R_sh
```

```
16 I_a = I_L - I_sh
17 E_b = V - I_a*R_a
18 N=E_b*60*A/(phi*P*Z)  //because E_b=  phi*P*N*Z/(60*
      A)
19
20 T_sh= P_out/(2*%pi*N/60) //Useful torque
21 printf('(i)Speed at full load = %.4f rpm',N)
22 printf('\n(ii)Useful torque = %.2f N-m',T_sh)
23 printf('\n\nAnswer mismatches due to improper
      approximation')
```

Scilab code Exa 2.36 TO DETERMINE MOTOR SPEED IF ADDITIONAL RESISTANCE IS INSERTED IN SERIES WITH ARMATURE CIRCUIT

```
1  clc,clear
2  printf('Example 2.36\n\n')
3
4  N_1=1000
5  I_1=50,I_a1=I_1
6  V=250
7  R_x=4.4, R_t=0.6    //R_t = R_a+R_se
8  E_b1=V - I_a1*(R_t)
9
10 //T (prop.)I_a^2  , T (prop.) N^2  .... hence N (
      prop.) I_a
11 //N (prop.) E_b /I_a
12 //combining both , E_b (prop.) I_a^2
13 //using E_b2 = V - I_a2*(R_a + R_se + R_x) and
      solving for I_a2 , we get 0.088 I_a2^2 +5 I_a2
      -250=0
14 p=[0.088 5 -250]
15 roots(p)
16 I_a2=ans(2) //root(1) is ignored as it is -ve
17 E_b2 = V - I_a2*(R_t + R_x)
18 N_2=N_1*(E_b2/E_b1)*(I_a1/I_a2)
```

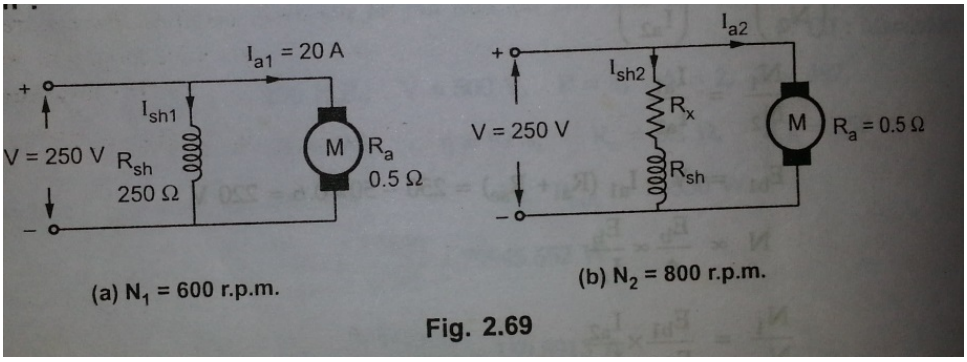


Figure 2.23: TO DETERMINE RESISTANCE TO BE INSERTED IN SHUNT FIELD CIRCUIT TO INCREASE THE SPEED

```
19 printf('Motor speed = %.2f r.p.m',N_2)
```

Scilab code Exa 2.37 TO DETERMINE RESISTANCE TO BE INSERTED IN SHUNT FIELD CIRCUIT TO INCREASE THE SPEED

```
1  clc,clear
2  printf('Example 2.37\n\n')
3
4  V=250, I_a1=20
5  R_sh= 250,R_a=0.5 //shunt field and armature
       resistance
6  I_sh1= V / R_sh
7  E_b1 = V - I_a1*R_a
8
9  //T (prop.) phi*I_a (prop.) I_sh*I_a
10 //since T_1 = T_2, I_sh2_I_a2 = I_sh1*I_a1
11 I_sh2_I_a2 = I_sh1*I_a1  //=20
12
13 //N (prop.) E_b/I_sh
14 //E_b1 = V - I_a1*R_a
```

```
15 //Solving further for I_a2, we get I_a2^2 -500 I_a2
     + 12800
16 p=[1 -500 12800]
17 roots(p)
18 I_a2=ans(2) //higher root is neglected
19 I_sh2= I_sh2_I_a2 / I_a2
20 R_x= (V / I_sh2) - R_sh //resistance to be inserted
     in shunt field
21 printf('Resistance to be inserted = %.4f ohms ',R_x)
```

Scilab code Exa 2.38 TO DETERMINE TORQUES BEFORE AND AFTER FIELD WEAKENING

```
1  clc,clear
2  printf('Example 2.38\n\n')
3
4  V=250, N_1=1000
5  I_L1=25
6  R_a=0.2, R_sh=250 //armature and shunt field
     resistance
7  V_brush= 1 //voltage drop due to brushes
8
9  I_sh1 = V/R_sh
10 I_a1= I_L1 - I_sh1
11 E_b1= V- I_a1*R_a - 2 *V_brush
12
13 //when loaded
14 I_L2=50
15 I_sh2=I_sh1   //as flux weakensby armature reaction,
     shunt field current remains same
16 I_a2= I_L2 - I_sh2
17 E_b2= V- I_a2*R_a - 2 *V_brush
18
19 phi2_by_phi1= 1- (3/100)  //weakens by 3 percent
20 N_2= N_1*(E_b2/E_b1)/ phi2_by_phi1   //N (prop.) E_b
```

```
    /phi
21 printf('New speed = %.3f rpm',N_2)
22 T_1= E_b1*I_a1/(2*%pi*N_1/60)
23 T_2= E_b2*I_a2/(2*%pi*N_2/60)
24 printf('\nTorque before field weakening = %.4f N-m',
    T_1)
25 printf('\nTorque after  field weakening = %.4f N-m',
    T_2)
```

Scilab code Exa 2.39 TO DETERMINE STALLING TORQUE AND TORQUES ON FULL LOAD AND DOUBLE FULL LOAD

```
1  clc,clear
2  printf('Example 2.39\n\n')
3
4  V=220
5  R_a=0.5, R_x=1 //armature resistance and extra
    resistance
6  N_FL=500 //full load speed in r.p.m
7  I_a_FL=30
8
9  //part(i)   Full load
10 E_b_FL= V- I_a_FL * R_a
11 //T (prop.) I_a... T is constant
12 I_a_dash_FL = I_a_FL
13 E_b_dash_FL =  V- I_a_dash_FL * (R_a+R_x)
14 //N (prop.) E_b/phi   (prop.) E_b
15 N_dash_FL = N_FL*(E_b_dash_FL/E_b_FL)
16 printf('(i)Speed at full load torque =%.4f r.p.m\n',
    N_dash_FL)
17
18 //part(ii)
19 T2_by_T1 = 2
20 I_a_dash_FL  =  I_a_FL  *(T2_by_T1)
21 E_b_dash_FL =  V- I_a_dash_FL * (R_a+R_x)
```

```
22 N_dash_FL = N_FL*(E_b_dash_FL/E_b_FL)
23 printf('(ii)Speed at double full load torque =%.3f r
      .p.m\n',N_dash_FL)
24
25 //part(iii) ...stalling
26 E_b=0 //as speed is zero in case of stalling torque
27 I_a_stall=(V-E_b)/(R_a+R_x)
28 T_FL = E_b_FL * I_a_FL/(2*%pi*N_FL/60)
29 T_stall = T_FL *(I_a_stall/ I_a_FL)
30 printf('(iii)Stalling torque = %.3f Nm',T_stall)
```

Scilab code Exa 2.40 TO DETERMINE SPEED OF MOTOR FULL LOAD TORQUE AND MULTIPLES OF FULL LOAD TORQUE

```
1 clc,clear
2 printf('Example 2.40\n\n')
3
4 V=230, I_a1=30
5 R_a=0.4,R_x=1.1//armature resistance and extra
      resistance
6 N_1=500
7
8 //part(i)
9 E_b1= V - I_a1*R_a
10 I_a2 = I_a1 //I_a is constant as T, phi are constant
11 E_b2= V - I_a2*(R_a+R_x)
12 N_2 = N_1 *(E_b2/E_b1) //Because N (prop.) E_b/phi (
      prop.) E_b
13 printf('(i)Speed at full load torque = %.3f r.p.m\n'
      ,N_2)
14
15 //part(ii)
16 T2_by_T1=1.5
17 I_a2= I_a1 * T2_by_T1
18 E_b2= V - I_a2*(R_a+R_x)
```

```
19 N_2 = N_1 *(E_b2/E_b1) //Because N (prop.) E_b/phi (
     prop.) E_b
20 printf('(ii)Speed at 1.5 times full load torque = %
     .3f r.p.m\n',N_2)
```

Chapter 3

Testing of DC Macines

Scilab code Exa 3.1 TO DETERMINE CERTAIN QUANTITIES RELATED TO WAVE CONNECTED SHUNT MOTOR

```
1  clc,clear
2  printf('Example 3.1\n\n')
3
4  Pole=6
5  V=500
6  A=2 //because of wave wound armature
7  Z=1200 //number of armature conductors
8  phi=20*10^-3 //useful flux per pole
9  Ra=0.5 ,Rsh=250  //armature and field resistance
10 Il=20 //current drawn from supply
11 mechanical_losses=900
12 Ish=V/Rsh
13 Ia=Il-Ish
14 Eb=V-Ia*Ra  //because V=Eb+Ia*Ra
15 N=Eb*60*A/(phi*Pole*Z)    //Eb=phi*Pole*N*Z/(60*A)
16
17 P_m=Eb*Ia  //Electrical equivalent of mechanical
     power
18 omega=2*%pi*N/60
19 Tg=P_m/omega
```

```
20
21 P_out=P_m-mechanical_losses
22 T_sh=P_out/omega  //Useful torque
23 P_in=V*Il
24 percentage_efficiency=100*P_out/P_in
25
26 printf('Speed developed is %.3f r.p.m\nTorque
      developed is %.2f N-m\n\n(i) Useful torque is %
      .2f N-m\n(ii) Efficiency is %.2f percent',N,Tg,
      T_sh,percentage_efficiency)
```

Scilab code Exa 3.2 TO DETERMINE FULL LOAD FULL LOAD OUTPUT AND EFFICIENCY

```
1 clc,clear
2 printf('Example 3.2\n\n')
3
4 //no load
5 I_noload=2.5 //No load current
6 V=440
7 R_a=1.2,R_sh=550//resistance of armature and shunt
      field windings
8 no_load_input=V*I_noload
9
10 I_sh=V/R_sh
11 I_a_noload=I_noload-I_sh
12 no_load_armature_copper=(I_a_noload^2)*R_a
13 constant_losses=no_load_input-
      no_load_armature_copper
14
15 //full load
16 I_fullload=32
17 I_a_fullload=I_fullload-I_sh
18 full_load_armature_coppe=(I_a_fullload^2)*R_a
19 total_losses=full_load_armature_coppe+
```

```
    constant_losses
20 full_load_motor_input=V*I_fullload
21 full_load_motor_output=full_load_motor_input-
    total_losses
22 efficiency_at_full_load=full_load_motor_output*100/
    full_load_motor_input
23
24 printf('Full load motor output is %.2f W\nEfficiency
    of motor at full-load is %.2f percent',
    full_load_motor_output,efficiency_at_full_load)
```

Scilab code Exa 3.3 TO ESTIMATE FULL LOAD CURRENT AND EFFICIENCY

```
1  clc,clear
2  printf('Example 3.3\n\n')
3
4  //no load
5  I=14 //input current
6  V=230
7  power_output_FL = 45*10^3
8  power_input=V*I
9  I_sh=2.55 //field current
10 R_a=0.032 //armature resistance
11 I_a=I-I_sh
12 cu_loss_NL = I_a^2*R_a //no load copper loss
13 brush_loss=2*I_a
14 constant_loss= power_input - cu_loss_NL - brush_loss
15
16 //full load
17
18 //I=I_a+ 2.55
19 //Motor input= Motor output + constant loss + brush
    loss + cu loss
20 // solving for I_a , I_a^2 - 7125 I_a + 1487700.3 =0
```

```
21 p=[1 -7125 1487700.3]
22 roots(p)
23 I_a=ans(2) //ignoring second root as its too large
24 I=I_a+I_sh
25 printf('Full load current is %.2f A\n',I)
26 power_input=V*I
27 eta=100*(power_output_FL/power_input)
28 printf('Efficiency at full load is %.2f percent',eta
   )
```

Scilab code Exa 3.4 TO CALCULATE FULL LOAD EFFICIENCY OF DC SHUNT MOTOR

```
1 clc,clear
2 printf('Example 3.4\n\n')
3
4 W1=9.1  //Tension on tight side
5 W2=0.8  //Tension on slack side
6 I=10    //Total_current
7 V=110   //Supply voltage
8 R=7.5/100   //Radius of p-ulley in metres
9 N=1320     //speed in r.p.m
10 T_sh=(W1-W2)*9.81*R   //9.81 is the accelration due
   to gravity
11 omega=(2*%pi*N/60)
12 P_out=T_sh*omega
13 P_in=V*I
14
15 efficiency=100*P_out/P_in
16 printf('Full load Efficiency is %.2f percent',
   efficiency)
```

Scilab code Exa 3.5 TO FIND THE EFFICIENCY OF MOTOR

```
clc,clear
printf('Example 3.5\n\n')

V=250
I_av=10
V_av=(240+220)/2 //average voltage across load
W_dash=V_av*I_av  //Power absorbed
t1=25,t2=6
R_sh=200,R_a=0.3//resistance of field winding and
   armature

W=W_dash*t2/(t1-t2)   //Stray Losses
I_l=25 //Input current
I_sh=V/R_sh //current through field winding

I_a=I_l-I_sh   //Armature current
arm_cu_loss=R_a*I_a^2  //Armature copper losses
sh_cu_loss=R_sh*I_sh^2 // Shunt copper loss

Total_losses= arm_cu_loss + sh_cu_loss + W
Motor_input=V*I_l
Output=Motor_input- Total_losses
efficiency=Output*100/Motor_input
printf('Efficiency as motor at 25 A and 250 V is %.2
   f percent',efficiency)
```

Scilab code Exa 3.6 TO DETERMINE THE EFFICIENCY OF MACHINES

```
clc,clear
printf('Example 3.6\n\n')

I_a=37,I_sh=0.85//armature and field current for
   motor
```

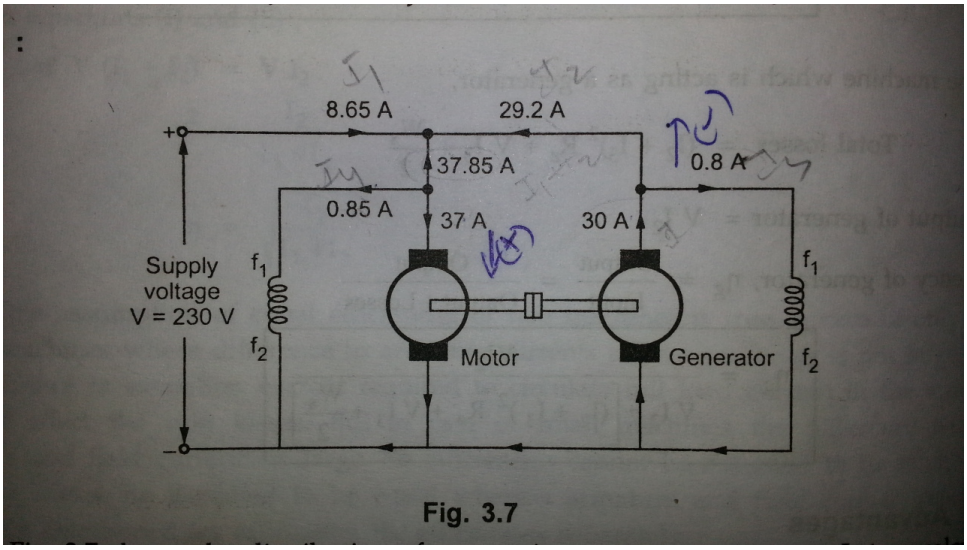


Figure 3.1: TO DETERMINE THE EFFICIENCY OF MACHINES

```
5  V=230
6  R_a=0.33 //armature resistance
7
8  I_a_g=30,I_sh_g=0.8//armature and field current for
      generator
9
10 //for motor
11 arm_cu_loss= I_a^2* R_a    //armature copper losses
12 field_cu_loss=V*I_sh  //field copper loss
13 total_cu_loss= field_cu_loss + arm_cu_loss //total
      copper loss
14
15 //for generator
16 arm_cu_loss_g= I_a_g^2* R_a    //armature copper
      losses
17 field_cu_loss_g=V*I_sh_g  //field copper loss
18 total_cu_loss_g= field_cu_loss_g + arm_cu_loss_g //
      total copper loss
19
20 //for motor-generator set
```

```
21 total_cu_loss_set=   total_cu_loss_g + total_cu_loss
22 P_supply=V*(I_a - I_a_g + I_sh+ I_sh_g ) //power
      taken from supply
23 stray_loss= P_supply - (total_cu_loss_g +
      total_cu_loss)
24 stray_loss_each=  stray_loss/2   //stray loss for
      each machine
25
26 //efficiency of motor
27 motor_input = V*(I_a+I_sh)
28 motor_output =  motor_input - (stray_loss_each +
      total_cu_loss)
29 eta_m=  100* motor_output/motor_input //efficiency
      of motor
30 printf('Efficiency of motor is %.2f percent \n',
      eta_m)
31 //efficiency of generator
32 generator_input = motor_output   //output of motor
      is input of generator
33 generator_output =  generator_input - (
      stray_loss_each + total_cu_loss_g)
34 eta_g=  100* generator_output/generator_input //
      efficiency of generator
35 printf('Efficiency of generator is %.2f percent \n',
      eta_g)
```

Scilab code Exa 3.7 TO FIND EFFICIENCY OF EACH MACHINE

```
1 clc,clear
2 printf('Example 3.7\n\n')
3
4 I_1=40 //motor input current
5 V=200  //voltage across armature
```

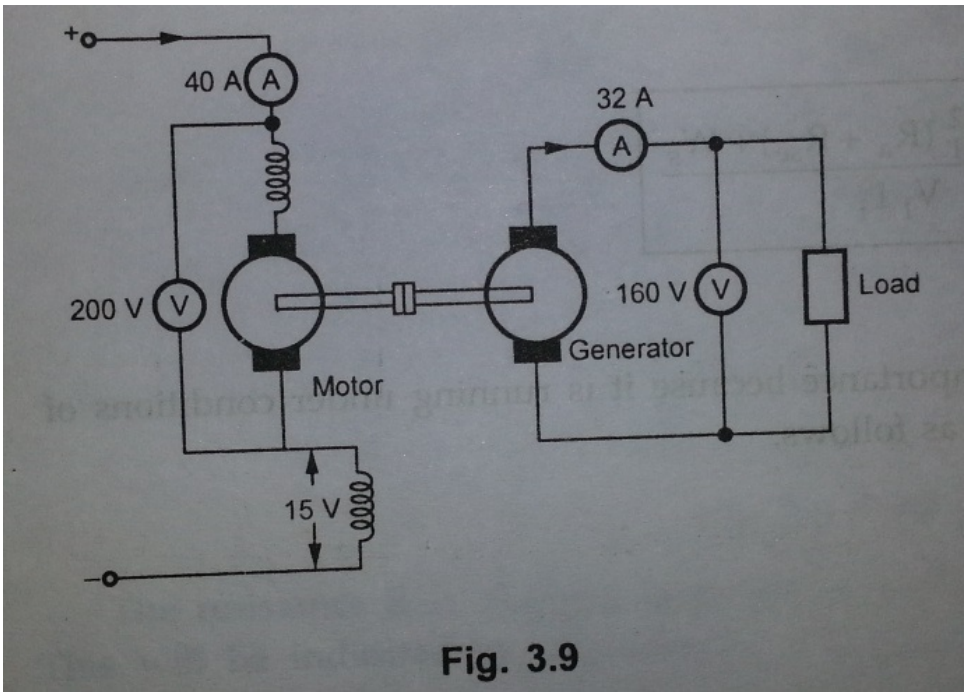


Figure 3.2: TO FIND EFFICIENCY OF EACH MACHINE

```
I_2=32  //load current
V_2=160  //voltage across generator
V_f=15 //voltage drop across field windings
total_input = (V+V_f)*I_1
Output=V_2*I_2
total_losses = total_input-Output   //total losses in 2 machines

R_se=V_f/I_1   //series field resistance
R_a=0.4 // armature field resistance

total_cu_loss=(R_a + 2*R_se) * I_1^2 + I_2^2*R_a //total copper loss
stray_losses =  total_losses -  total_cu_loss
stray_losses_each =stray_losses /2 //stray losses for each machine

//for motor
motor_input= V*I_1
arm_cu_loss= (R_a + R_se)*I_1*I_1 //armature copper loss
total_losses_motor = arm_cu_loss + stray_losses_each
motor_output= motor_input- total_losses_motor
eta_m=100*motor_output/motor_input //efficiency of motor
printf('Efficiency of motor is %.2f percent \n', eta_m)
//for generator
arm_cu_loss_gen=R_a*I_2^2 //armature copper loss
series_field_cu_loss = V_f*I_1 //series field copper loss
total_losses_gen= arm_cu_loss_gen + series_field_cu_loss + stray_losses_each
generator_input = total_losses_gen+ Output
eta_gen=100*Output/generator_input //efficiency of generator
printf('Efficiency of generator is %.2f percent',
```

```
    eta_gen)
34 printf('\n\nAnswer dont match because Output-of-
    generator is taken as 5220 for calculation while
    its should have been 5120')
```

Scilab code Exa 3.8 TO DETERMINE EFFICIENCY WHEN MOTOR DRAWS 100 A CURRENT

```
1  clc,clear
2  printf('Example 3.8\n\n')
3
4  V=500
5  Io=5 //no load current
6  R_a=0.5,R_sh=250//resistance of armature and field
       circuits
7  I=100 //current at unknown efficiency
8
9  P_in_NL=V*Io  //no load input
10 I_sh=V/R_sh
11
12 Iao=Io-I_sh
13 arm_cu_loss_no_load=R_a*Iao^2  //No load armature
       copper loss
14 constant_losses= P_in_NL- arm_cu_loss_no_load
15
16 I_a=I-I_sh
17 arm_cu_loss= R_a*I_a^2  //New armature copper loss
18
19 Total_loss=arm_cu_loss + constant_losses
20 P_in=V*I
21 efficiency=(P_in-Total_loss)*100/P_in //required
       efficiency
22 printf('Efficiency is %.3f percent when motor takes
    %.0f A current',efficiency,I)
```

Scilab code Exa 3.9 TO DETERMINE EFFICIENCY AND PERCENTAGE CHANGE IN SPEED

```
clc,clear
printf('Example 3.9\n\n')

V=500
I_NL=5  //no load current
P_in_NL = V* I_NL   //no load input
R_a=0.22,R_sh=250 //resistance of armature and shunt
    field winding
I_sh=V/R_sh
I_a_NL=I_NL-I_sh //armature current no load
arm_cu_loss_NL = R_a*I_a_NL^2  //No-load armature
   copper loss
constant_loss =  P_in_NL -  arm_cu_loss_NL

//at 100 A current
I=100
I_a = I - I_sh
arm_cu_loss =  R_a*I_a^2 //armature copper loss
total_loss = arm_cu_loss +  constant_loss
motor_input = V*I
motor_output =  motor_input- total_loss
eta_m= 100*motor_output/motor_input  //motor
   efficiency
printf('Part(a)\nEfficiency of motor when it takes
   100 A current and loaded is %.2f percent\n',eta_m
   )

//part(b)
E_b_NL = V - I_a_NL*R_a   //back emf at no load

E_b = V- I_a*R_a  //back EMF at 100 A current
```

```
27 //Since E_b is proportional to N and using
     componendo dividendo
28 delta_speed= 100*((E_b_NL-E_b)/E_b)
29 printf('Part(b)\nPercentage speed in speed is %.3f
     percent\n\n',delta_speed)
30
31 printf('Note that the following were assumptions
     made\n')
32 printf('(i) Due to heating , resistance of shunt
     field winding will be increased which will reduce
      the shunt field current.This will decrease the
     flux which is neglected\n')
33 printf('(ii) Though the motor speed is changing from
      no load to given load , the mechanical losses
     are assumed to be cosnstant\n')
34 printf('(iii)The effect of armature reaction aon
     main pole flux and its effect on iron loss is
     neglected')
```

Scilab code Exa 3.10 TO DETERMINE EFFICIENCY AND SPEED WHEN MOTOR DRAWS CERTAIN CURRENT

```
1 clc,clear
2 printf('Example 3.10\n\n')
3
4 motor_output_FL =15000 //full load motor output
5 V=250,R_sh=100
6
7 //at 80 % of full load
8 motor_output_FL_dash=(80/100)*motor_output_FL  //80
     percent of full load output
9 eta=90/100 //efficiency
10 motor_input=motor_output_FL_dash/eta
11 total_losses = motor_input - motor_output_FL_dash
     //at 80 % of full load
```

```
12 //at maximum efficiency , variable losses = constant
       losses
13 constant_losses= total_losses/2
14 variable_losses= constant_losses
15 I= motor_input/V //line current at 80% load
16 I_sh= V/ R_sh
17 I_a= I- I_sh
18 //since armature copper loss =R_a*I_a^2
19 R_a=variable_losses/I_a^2
20 E_b1=V-I_a*R_a //motor back EMF at 80% of full load
21 N_1=750 // corresponding speed is given as 750 rpm
22
23 //When motor current is 80 A
24 I=80
25 I_a=I-I_sh
26 arm_cu_losses= R_a*I_a^2  //armature copper loss
27 total_losses =  arm_cu_losses + constant_losses
28 motor_input= V*I
29 motor_output = motor_input- total_losses
30 eta=100*motor_output/motor_input //efficiency of
       motor
31 printf('Efficiency of motor is %.2f percent when
       motor draws 80A current',eta)
32 E_b2=V-I_a*R_a //motor back EMF at 80% of full load
33 N_2=N_1*(E_b2/E_b1)  //because E_b is proportional
       to N
34 printf('\nand Speed is %.2f r.p.m',N_2)
```

Scilab code Exa 3.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 250 V DC HUNT MOTOR

```
1 clc,clear
2 printf('Example 3.11\n\n')
3
4 V=250
```

```
5  R_sh=166.67,R_a=0.15 //resistance of shunt field
       winding and armature
6  N_0=1280 //in rpm
7  I_L1=67 //current drawn on full load
8  I_sh= V/ R_sh
9  I_a1=I_L1 - I_sh
10 E_b1=V-I_a1*R_a
11
12 //on no load
13 I_L0=6.5
14 I_a0= I_L0 - I_sh
15 E_b0=V-I_a0*R_a
16
17 //part (i)
18 //using speed equation  N is proportional to E_b
19 N_1=N_0*(E_b1/E_b0)
20 printf('(i)Full load speed is %.3f r.p.m\n',N_1)
21
22 //part (ii)
23 speed_regulation=100*((N_0-N_1)/N_1)
24 printf('(ii)Speed regulation is %.2f percent \n',
       speed_regulation)
25
26 //part (iii)
27 stray_losses = E_b0*I_a0  //mechanical power
       developed on no load
28 power_developed_FL= E_b1*I_a1
29 shaft_output_FL= power_developed_FL - stray_losses
30 hp_rating= shaft_output_FL/746 //in horse power
31 printf('(iii)H.P rating of the machine Is %.2f H.P\n
       ',hp_rating)
32
33 //part (iv)
34 power_input=V*I_L1
35 eta=100*(shaft_output_FL/power_input)  //efficiency
       at full load
36 printf('(iv)Efficiency at full load is %.2f percent\
       n',eta)
```

Scilab code Exa 3.12 TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V SHUNT MOTOR

```
1 clc,clear
2 printf('Example 3.12\n\n')
3
4 V=200
5 R_sh=240,R_a=0.1 //resistance of shunt field winding
    and armature
6 rotational_loss=236
7 I_L_FL=9.8  //full load line current
8 N=1450
9 I_sh=V/R_sh
10 I_a_FL = I_L_FL - I_sh
11 E_b= V- I_a_FL * R_a
12
13 //part(i)
14 gross_mech_P_dev= E_b*I_a_FL  //gross mechanical
   power developed
15 mech_P_dev= gross_mech_P_dev - rotational_loss //
   mechanical power developed
16 printf('(i)Gross mechanical power developed is %.2f
   W\n',gross_mech_P_dev )
17 printf('   Mechanical power developed is %.2f W\n',
   mech_P_dev )
18
19 //part(ii)
20 P_out=mech_P_dev
21 printf('(ii)The power output is %.2f W\n',P_out)
22
23 //part(iii)
24 T_sh=P_out*60/(2*%pi*N)
25 T_L=T_sh
26 printf('(iii)Load torque is %.2f N-m\n',T_L)
```

```
27
28 //part(iv)
29 P_in=V*I_L_FL
30 eta=100*P_out/P_in
31 printf('(iv)Efficiency at full load is %.2f percent\
   n',eta)
```

Scilab code Exa 3.13 TO DETERMINE CERTAIN QUANTITIES RELATED TO 240 V DC SHUNT MOTOR

```
1  clc,clear
2  printf('Example 3.13\n\n')
3
4  V=240
5  P_out=25*735.5  //output power in watts
6  R_a=0.14,R_sh=80 //resistance of armature and shunt
      field winding
7  brush_drop=1 //voltage drop across brush
8  I_L_FL=95 //input current at full load
9  I_sh=V/R_sh
10 I_a_FL = I_L_FL - I_sh //armature current at full
      load
11
12 arm_cu_loss_FL = R_a*I_a_FL^2  //full load armature
      copper loss
13 field_cu_loss= R_sh*I_sh^2  //field copper loss
14 printf('(i)Armature and field copper losses are %.2f
       W and %.0f W respectively\n',arm_cu_loss_FL,
      field_cu_loss)
15 brush_cu_loss= 2*brush_drop*I_a_FL //brush contact
      copper loss
16 printf('(ii)Brush contact copper loss is %.0f W\n',
      brush_cu_loss)
17 E_b=V-I_a_FL*R_a - 2*brush_drop //back emf
18 gross_mech_P_dev= E_b*I_a_FL  //gross mechanical
```

```
    power developed
19 IFW_losses = gross_mech_P_dev - P_out  //iron
    friction and windage losses
20 printf('(iii)Core plus mechanical losses = %.1f W\n'
    ,IFW_losses+field_cu_loss+arm_cu_loss_FL)
21 eta=100*(P_out/(P_out + IFW_losses+ brush_cu_loss+
    field_cu_loss+arm_cu_loss_FL  ))
22 printf('(iv)Efficiency is %.2f percent',eta)
```

Scilab code Exa 3.14 TO DETERMINE FULL LOAD OUTPUT AND EFFICIENCY

```
1  clc,clear
2  printf('Example 3.14\n\n')
3
4  I=5  //no load current
5  V=500
6  R_sh=250,R_a=0.5 //resistance of shunt field winding
    and armature
7  motor_input_NL = V*I
8  I_sh=V/R_sh
9  I_a=I-I_sh
10 arm_cu_loss_NL = R_a*I_a^2  //no load armature
    copper loss
11 constant_loss = motor_input_NL - arm_cu_loss_NL
12 I_FL=50, I_a_FL = I_FL - I_sh   //currents at full
    load
13 arm_cu_loss_FL =  R_a*I_a_FL^2 //full load armature
    copper loss
14
15 total_loss= constant_loss + arm_cu_loss_FL
16 motor_input=V*I_FL
17 motor_output_FL= motor_input - total_loss
18 printf('Required output power is %.3f kW\n',
    motor_output_FL/1000)
```

```
19 eta=100*(motor_output_FL/motor_input) //full load
     efficiency
20 printf('Full load efficiency of motor with 50A
     current is %.2f percent ',eta)
```

Scilab code Exa 3.15 TO CALCULATE MACHINE EFFICIENCY WHEN OPERATING AS A GENERATOR

```
1  clc,clear
2  printf('Example 3.15\n\n')
3
4  V=250
5  R_sh=275,R_a=0.8 //resistance of shunt field and
     amature
6  I_L0=3.91 //load current
7  I_sh=V/R_sh
8  I_a0= I_L0 - I_sh
9  constant_losses= V*I_L0 -R_a*(I_a0)^2
10
11 //as a generator
12 P_out=10*10^3
13 I_L=P_out/V
14 I_a = I_L + I_sh
15 field_cu_loss=R_sh*(I_sh)^2  //field copper loss
16 arm_cu_loss= R_a*(I_a)^2  //armature copper loss
17 eta_gen = 100 *(P_out/(P_out+constant_losses +
     field_cu_loss+ arm_cu_loss))  //efficiency as
     generator
18 printf('Efficiency as a generator = %.2f percent\n',
     eta_gen)
19
20 //as a motor
21 P_in=10*10^3  //at V=250
22 I_L=P_in/V
23 I_a=I_L - I_sh
```

```
24 field_cu_loss=R_sh*(I_sh)^2  //field copper loss
25 arm_cu_loss= R_a*(I_a)^2  //armature copper loss
26 eta_m = 100 *((P_in-(constant_losses + field_cu_loss
      + arm_cu_loss))/(P_in)) //efficiency as motor
27 printf('Efficiency as a motor = %.2f percent',eta_m)
```

Scilab code Exa 3.16 TO DETERMINE FULL LOAD OUTPUT POWER EFFICIENCY AND PERCENTAGE CHANGE IN SPEED

```
1  clc,clear
2  printf('Example 3.16\n\n')
3
4  I=4 //no load current in amperes
5  V=500
6  motor_input_no_load=I*V   //no load motor input
7  R_a=0.5,R_sh=250//resistance of armature and shunt
      field resistnace
8  I_sh=V/R_sh
9
10 I_a=I-I_sh
11 arm_cu_loss_noload=R_a*I_a^2  //No-load armature
      copper losses
12 constant_loss=motor_input_no_load -
      arm_cu_loss_noload
13 I_FL=40,I_aFL=I_FL- I_a  //full load currents
14 arm_cu_loss_fulload=R_a*I_aFL^2 //Full-load armature
       copper losses
15 Total_loss=arm_cu_loss_fulload + constant_loss
16
17 motor_input=V*I_FL
18 motor_output_fullload=motor_input - Total_loss
19 printf('Output power at full-load is %.0f W',
      motor_output_fullload)
20 efficiency= motor_output_fullload*100/motor_input //
      motor efficiency
```

```
21 printf('\nEfficiency at full-load is %.1f percent',
      efficiency)
22
23 E_bNL=V-I_a*R_a
24 E_bFL=V-I_FL*R_a
25
26 //E_b =N*phi
27 //E_bNL/E_bFL=N_NL/N_FL
28 //applying rules of componendo and dividendo
29 //change_in _speed=(N_NL - N_FL)/N_FL=(E_bNL - E_bFL
      )/E_bFl
30
31 change_in_speed=100*(E_bNL - E_bFL)/E_bFL
32 printf('\npercentage change in speed from no load to
       full load is %.3f percent',change_in_speed)
```

Scilab code Exa 3.17 TO DETERMINE EFFICIENCY AND PERCENTAGE CHANGE IN SPEED OF A SHUNT MOTOR

```
1 clc,clear
2 printf('Example 3.17\n\n')
3
4 V=500
5 R_a=0.22,R_sh= 250//armature resistance and shunt
      field resistance
6 I=5 //no load current
7 motor_input_NL=V*I   //no load motor input
8 I_sh=V/R_sh
9 I_a_NL= I- I_sh //no load armature current
10 arm_cu_loss_NL = R_a*I_a_NL^2 //no load armature
      copper loss
11 constant_loss = motor_input_NL - arm_cu_loss_NL
12
13 //When motor draws 100 A current
14 I=100
```

```
15 I_a = I - I_sh
16 arm_cu_loss  =R_a* I_a^2  //armature copper loss
17 total_losses =   arm_cu_loss +  constant_loss
18 motor_input = V*I
19 motor_output = motor_input -total_losses
20 eta_m=100*(motor_output/motor_input) //motor
      efficiency
21 printf('(i)Efficiency of motor at 100 A current is %
      .2f percent \n',eta_m)
22
23 //part(b)
24 E_b_NL= V- I_a_NL*R_a //back emf at no load
25 E_b= V- I_a*R_a //back emf at 100 A
26 //E_b is proportional to N.. and using componendo
      dividendo
27 speed_change= 100*((E_b_NL - E_b)/E_b)
28 printf('(ii)Percentage change in speed = %.3f
      percent\n\n',speed_change)
29
30
31 printf('Note that the following were assumptions
      made\n')
32 printf('(i) Due to heating , resistance of shunt
      field winding will be increased which will reduce
       the shunt field current.This will decrease the
      flux which is neglected\n')
33 printf('(ii) Though the motor speed is changing from
       no load to given load , the mechanical losses
      are assumed to be cosnstant\n')
34 printf('(iii)The effect of armature reaction aon
      main pole flux and its effect on iron loss is
      neglected')
```

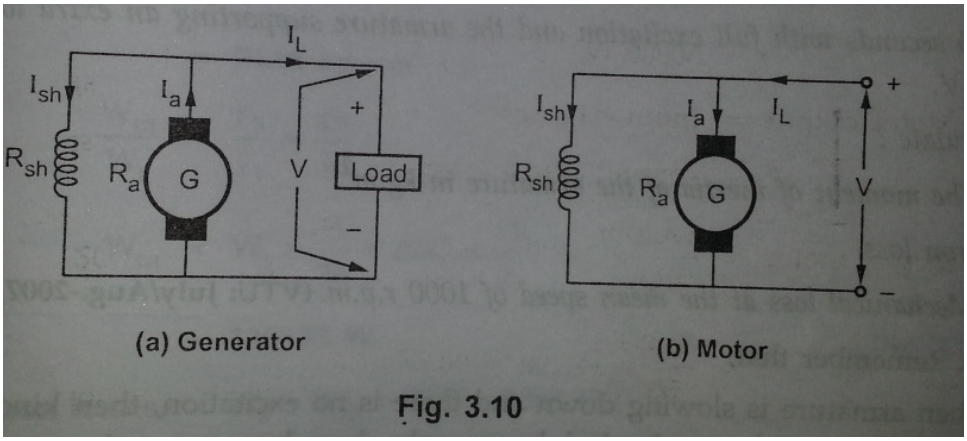


Fig. 3.10

Figure 3.3: TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V DC SHUNT MOTOR

Scilab code Exa 3.18 TO DETERMINE CERTAIN QUANTITIES RELATED TO 200 V DC SHUNT MOTOR

```
1  clc,clear
2  printf('Example 3.18\n\n')
3
4  V=200,I_L=40,
5  R_sh=200,R_a=0.2 //shunt field winding and armature
       resistance
6  //case(a) : As a generator
7  P_out_g=V*I_L //output poewr as generator
8  I_sh=V/R_sh
9  I_a = I_L + I_sh
10 E = V + I_a*R_a
11 P_a_g=E*I_a  //power developed in armature
12 P_cu_g= R_a*I_a^2 + R_sh*I_sh^2 //copper loss as
       genrator
13 printf('(i)Output as generator is %.0f kW\n',P_out_g
       /1000)
14
15 //case(b) : As a motor
16 P_in_m=V*I_L //input power as motor
```

```
17 I_sh=V/R_sh
18 I_a = I_L - I_sh
19 E_b = V - I_a*R_a
20 P_a_m=E_b*I_a  //power developed in armature
21 P_cu_m= R_a*I_a^2 + R_sh*I_sh^2//copper loss as
      motor
22 printf('(ii)Input as motor is %.0f kW\n', P_in_m
      /1000)
23 printf('\n(iii)Power developed in Armature:\n%.4f kW
       for generator\n%.4f kW for motor\n',P_a_g/1000,
      P_a_m/1000)
24 printf('\n(iv)Copper losses:\n%.1f W for generator\
      n%.1f W for motor',P_cu_g,P_cu_m)
```

Scilab code Exa 3.19 TO DETERMINE CERTAIN QUANTITIES AFTER PERFORMING RETARDATION TEST ON DC MACHINE

```
1 clc,clear
2 printf('Example 3.19\n\n')
3
4 V=219,I=10
5 dN=1030 - 970 //given
6 t_1=36  //time with no excitation
7 t_2=9// time with full excitation and armature
      supporting an extra load of 10 A at 219 V
8 t_3=15 //time with full exciation
9 W_dash = V*I  //additioanl loss when armature is
      suddenly connected to loads
10 W_s= W_dash *(t_2/(t_3-t_2)) //total stray losses
11 N=1000  //speed in rpm
12 //Using W_s = (2*pi/60)^2 * I *N *dN / t_3 where W_s
       is stray losses
13 I= W_s*(t_3/dN)*(30/%pi)^2/N   //moment of inertia
14 W_m=W_s*(t_3/t_1) //mechanical losses
15 iron_losses= W_s - W_m
```

```
16
17 printf('(i)The moment of inertia of armature is %.2f
       kg-m^2\n',I)
18 printf('(ii)Iron loss= %.2f W\n',iron_losses)
19 printf('(iii)Mechanical losses at 1000 rpm mean
     speed is %.2f W',W_m)
20
21 printf('\n\nNoteworthy points:\n(1)When armature is
     slowing down and there is no excitation,then
     kinetic energy is used to overcome mechanical
     losses only.Iron losses are absent as excitation
     is absent\n(2)When excitation is given, kinetic
     energy is used to overcome both mechanical as
     well as iron losses.Total called stray losses.\n
     (3)If moment of inertia is in kg-m^2,then loss of
       energy is in watts')
```

Scilab code Exa 3.20 TO DETERMINE CERTAIN QUANTITIES AFTER PERFORMING RETARDATION TEST ON DC MACHINE

```
1 clc,clear
2 printf('Example 3.20\n\n')
3
4 V=225,I=10
5 dN=1030 - 970 //given
6 t_1=40  //time with no excitation
7 t_2=9// time with full excitation and armature
     supporting an extra load of 10 A at 219 V
8 t_3=20 //time with full exciation
9
10 W_dash = V*I //additional loss
11 W_s= W_dash *(t_2/(t_3-t_2)) //total stray losses
12 N=1000//Speed in rpm
13 //Using W_s = (2*pi/60)^2 * I *N *dN / t_3 where W_s
       is stray losses
```

```
14 I= W_s*(t_3/dN)*(30/%pi)^2/N    //moment of inertia
15 W_m=W_s*(t_3/t_1)    //mechanical losses
16 iron_losses=W_s - W_m
17
18 printf('(i)The moment of inertia of armature is %.2f
       kg-m^2\n',I)
19 printf('(ii)Iron loss= %.2f W\n',iron_losses)
20 printf('(iii)Mechanical losses at 1000 rpm mean
      speed is %.2f W',W_m)
21
22 printf('\n\nNoteworthy points:\n(1)When there is no
      excitation and armature is slowed down , its K.E.
       is used to overcome mechanical mechanical losses
       only since there will be no iron loss as there
      is no flux.\n(2)When there is excitation provided
       then K.E. is used to supply mechanical as well
      as iron losses together called stray losses ')
```

Scilab code Exa 3.21 TO DETERMINE EFFICIENCY WHEN MACHINE IS OPERATED AS MOTOR

```
1 clc,clear
2 printf('Example 3.21\n\n')
3
4 V_avg = (220+190)/2  //average voltage across load
5 I_avg=12,R_a=0.5,R_sh=250
6 W_dash=V_avg*I_avg  //power absorbed
7 t_1=30,t_2=5
8 W=W_dash*(t_2/(t_1-t_2))
9 V=250,I=22  //input current
10 I_sh = V/R_sh
11 I_a= I - I_sh
12 arm_cu_loss = R_a*I_a^2 //armature copper loss
13 shunt_field_cu_loss =  V*I_sh  //shunt field copper
      loss
```

```
14 total_losses= shunt_field_cu_loss + arm_cu_loss + W
15
16 machine_input = V*I
17 machine_output = machine_input - total_losses
18 eta_m=100*(machine_output /machine_input )  //
   efficiency when running as motor
19 printf('Efficiency of machine when opeating as motor
    taking current of 22A on 250V supply is \n%.1f
   percent',eta_m)
```

Scilab code Exa 3.22 TO DETERMINE STRAY LOSSES OF MOTOR

```
1 clc,clear
2 printf('Example 3.22\n\n')
3
4 I_avg=10
5 V_avg=(220+190)/2 //average voltage across load
6 W_dash = V_avg*I_avg  //power  absorbed
7 t_1=30,t_2=20
8 W=W_dash * (t_2/(t_1-t_2)) //stray losses
9 printf('Stray losses of motor is %.1f kW\n\n\n',W
   /1000)
10 printf('Answers mismatch because V_average is 205
   volts but it is taken as 220 volts in Power
   absorbed calculation')
```

Scilab code Exa 3.23 TO DETERMINE EFFICIENCY OF EACH OF THE 2 SHUNT MACHINES

```
1 clc,clear
2 printf('Example 3.23\n\n')
```

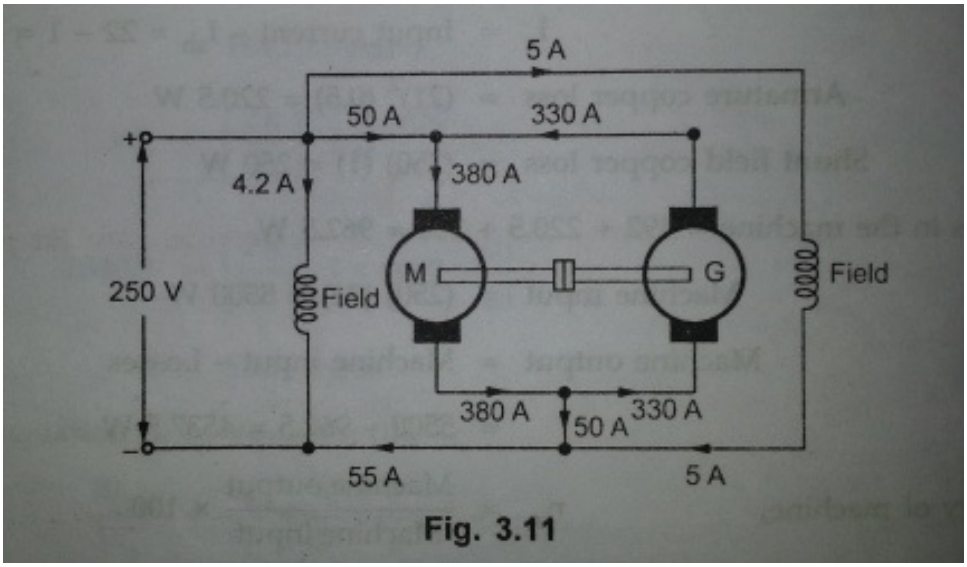


Figure 3.4: TO DETERMINE EFFICIENCY OF EACH OF THE 2 SHUNT MACHINES

```
3
4  I_a_g=330,I_a_m=380
5  R_a=0.02 //armature resistance
6  V=250,I=50
7  arm_cu_loss_g= R_a*I_a_g^2//armature copper loss for
       generator
8  arm_cu_loss_m= R_a*I_a_m^2//armature copper loss for
       motor
9  power_drawn=V*I
10 stray_losses = power_drawn - (arm_cu_loss_m +
     arm_cu_loss_g)
11 stray_losses_each = stray_losses/2 //stray losses
     for each machine
12
13 //for motor
14 I_sh_m=4.2   //Shunt current in case of motor
15 field_cu_loss_m=V*I_sh_m  //field copper loss in
     case of motor
16 total_loss = field_cu_loss_m + stray_losses_each  +
```

```
    arm_cu_loss_m
motor_input= V*(I_a_m+I_sh_m)
motor_output = motor_input - total_loss
eta_m = 100*(motor_output/motor_input)//motor
    efficiency
printf('Efficiency of motor is %.4f percent\n',eta_m
    )

//for generator
I_sh_g=5  //Shunt current in case of generator
field_cu_loss_g=V*I_sh_g //field copper loss in case
    of generator
total_loss = field_cu_loss_g + stray_losses_each  +
    arm_cu_loss_g
generator_output = V*I_a_g
generator_input= generator_output + total_loss
eta_g = 100*(generator_output/generator_input)//
    generator efficiency
printf('Efficiency of generator is %.4f percent\n',
    eta_g)
```

Scilab code Exa 3.24 TO DETERMINE EFFICIENCY OF MOTOR AND GENERATOR

```
clc,clear
printf('Example 3.24\n\n')

R_a=0.02 //armature resistance
V=250 //line voltage
I=50 //current taken from supply

//for generator
I_a_g=330,I_sh_g=5 //armature current and current
```

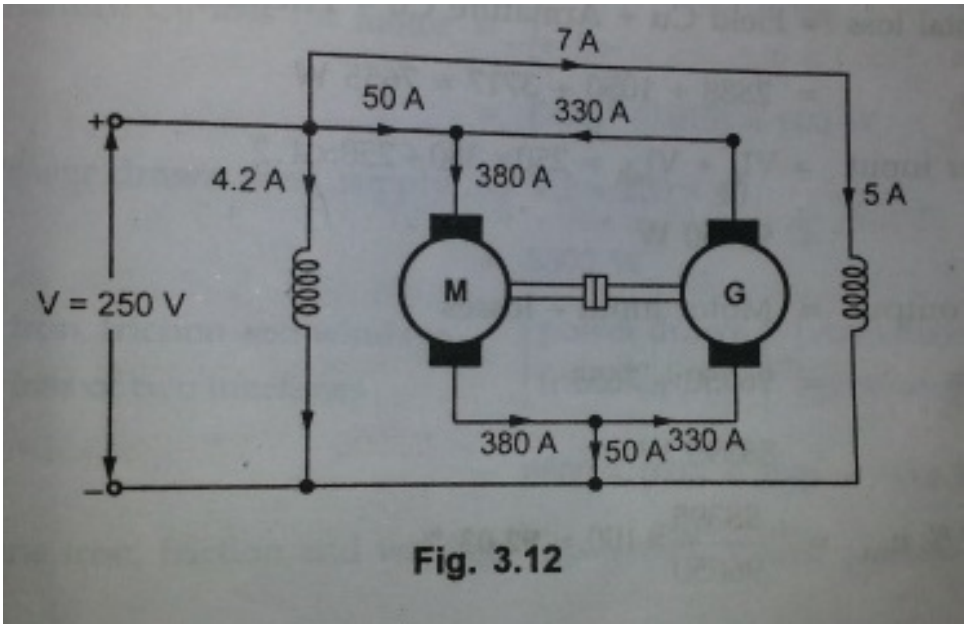


Figure 3.5: TO DETERMINE EFFICIENCY OF MOTOR AND GENERATOR

```
   through shunt field
10 arm_cu_loss_g = R_a*I_a_g^2//armature copper loss
   for generator
11 field_cu_loss_g= V*I_sh_g //field copper loss for
   generator
12
13 //for motor
14 I_a_m=380,I_sh_m=4.2 //armature current and current
   through shunt field
15 arm_cu_loss_m = R_a*I_a_m^2//armature copper loss
   for motor
16 field_cu_loss_m= V*I_sh_m //field copper loss for
   motor
17 power_drawn=V*I
18 IFW_losses = power_drawn - (arm_cu_loss_g +
   arm_cu_loss_m)  //Iron , friction and windage
   losses
```

```
19 IFW_losses_each= IFW_losses /2  // Iron , friction
   and windage losses for each machine
20
21 //for generator
22 total_loss_g = field_cu_loss_g + arm_cu_loss_g +
   IFW_losses_each
23 generator_output=V*I_a_g
24 generator_input = generator_output + total_loss_g
25 eta_g = 100*(generator_output/generator_input)//
   generator efficiency
26 printf('Efficiency of generator is %.4f percent\n',
   eta_g)
27
28 //for motor
29 total_loss_m= field_cu_loss_m + IFW_losses_each  +
   arm_cu_loss_m
30 motor_input=V*(I_a_m+I_sh_m)
31 motor_output = motor_input - total_loss_m
32 eta_m = 100*(motor_output/motor_input)//motor
   efficiency
33 printf('Efficiency of motor is %.4f percent\n',eta_m
   )
```

Scilab code Exa 3.25 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES

```
1 clc,clear
2 printf('Example 3.25\n\n')
3
4 V=220, I=40
5 I_a_g=160 ,I_a_m =200 //armature currents for
   generator and motor
6 I_sh_g=7 ,I_sh_m =6 //current through shunt field
```

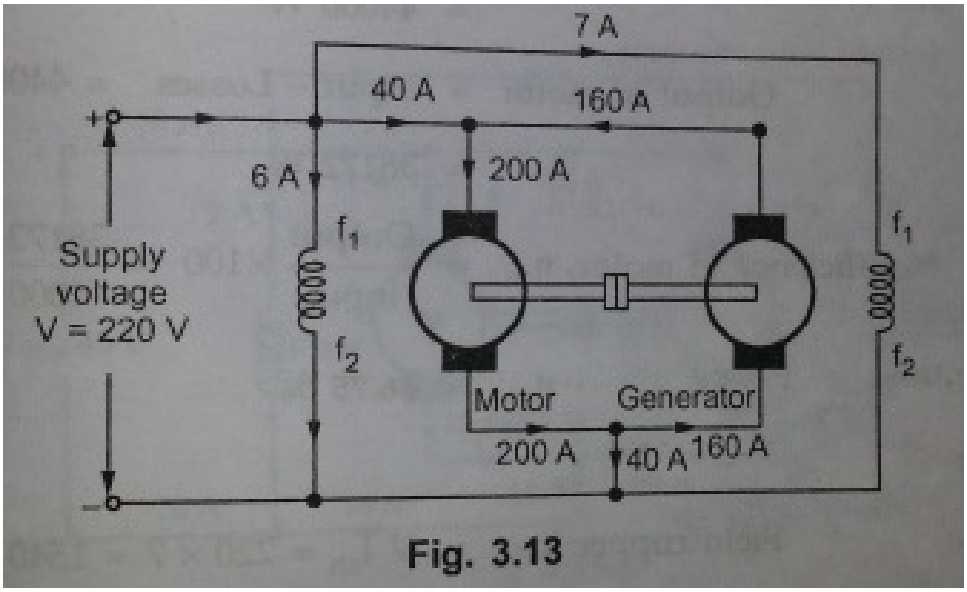


Figure 3.6: TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES

```
    for generator and motor
R_a=0.015 //armature resistance
arm_cu_loss_g = R_a*I_a_g^2  //armature copper loss
    for motor
arm_cu_loss_m = R_a*I_a_m^2  //armature copper loss
    for motor
power_drawn=V*I
IFW_losses = power_drawn - (arm_cu_loss_g +
    arm_cu_loss_m)  //Iron , friction and windage
    losses
IFW_losses_each= IFW_losses /2  // Iron , friction
    and windage losses for each machine

//for motor
field_cu_loss_m= V*I_sh_m //field copper loss for
    motor
total_loss_m= field_cu_loss_m + IFW_losses_each  +
    arm_cu_loss_m  //total losses in motor
```

```
17 motor_input=V * I_a_m
18 motor_output= motor_input - total_loss_m
19 eta_m = 100*(motor_output/motor_input)    //motor
      efficiency
20 printf('Efficiency of motor is %.4f percent\n',eta_m
      )
21
22 //for generator
23 field_cu_loss_g= V*I_sh_g //field copper loss for
      generator
24 total_loss_g = field_cu_loss_g + arm_cu_loss_g +
      IFW_losses_each  //total losses in generator
25 generator_output=V*I_a_g
26 generator_input = generator_output + total_loss_g
27 eta_g = 100*(generator_output/generator_input)//
      generator efficiency
28 printf('Efficiency of generator is %.4f percent\n',
      eta_g)
```

Scilab code Exa 3.26 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR ON FULL LOAD

```
1 clc,clear
2 printf('Example 3.26\n\n')
3
4 R_a=0.2 //armature resistance
5 V=240,I=16
6 I_a_g=60 , I_a_m=71 //armature currents for
      generator and motor
7 I_sh_g=3 , I_sh_m=2  //field current for generator
      and motor
8
9 //for generator
```

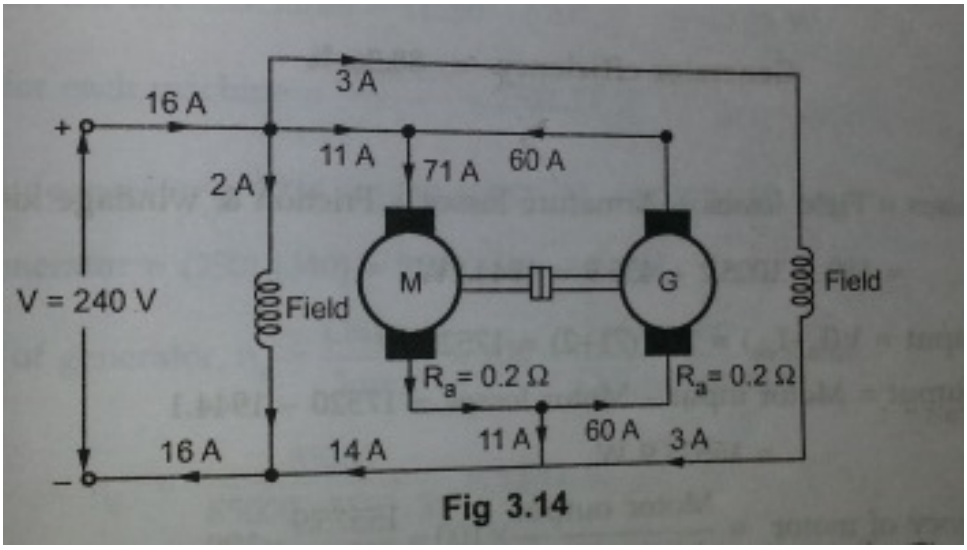


Figure 3.7: TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR ON FULL LOAD

```
10 arm_cu_loss_g = R_a*I_a_g^2//armature copper loss
     for generator
11 field_cu_loss_g= V*I_sh_g //field copper loss for
     generator
12
13 //for motor
14 arm_cu_loss_m = R_a*I_a_m^2//armature copper loss
     for motor
15 field_cu_loss_m= V*I_sh_m //field copper loss for
     motor
16 power_drawn=V*I
17 field_loss_total_g_m= field_cu_loss_m +
     field_cu_loss_g
18 arm_cu_loss_total_g_m = arm_cu_loss_m +
     arm_cu_loss_g
19 IFW_losses = power_drawn - (arm_cu_loss_total_g_m +
     field_loss_total_g_m)  //Iron , friction and
     windage losses
20 IFW_losses_each= IFW_losses /2  // Iron , friction
```

```
    and windage losses for each machine
21
22 //for generator
23 total_loss_g = field_cu_loss_g + arm_cu_loss_g +
    IFW_losses_each //total loss in generator
24 generator_output=V*I_a_g
25 generator_input = generator_output + total_loss_g
26 eta_g = 100*(generator_output/generator_input)//
    generator efficiency
27 printf('Efficiency of generator is %.4f percent\n',
    eta_g)
28
29 //for motor
30 total_loss_m= field_cu_loss_m + IFW_losses_each  +
    arm_cu_loss_m //total loss in motor
31 motor_input=V*(I_a_m+I_sh_m)
32 motor_output = motor_input - total_loss_m
33 eta_m = 100*(motor_output/motor_input)//motor
    efficiency
34 printf('Efficiency of motor is %.4f percent\n',eta_m
    )
```

Scilab code Exa 3.27 TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES

```
1 clc,clear
2 printf('Example 3.27\n\n')
3
4 R_a=0.015,V=250 //line voltage
5 I=45 //line current
6 I_a_m=385, I_sh_m=4 //armature and field currents
    for motor
7 I_a_g=340, I_sh_g=5 //armature and field currents
```

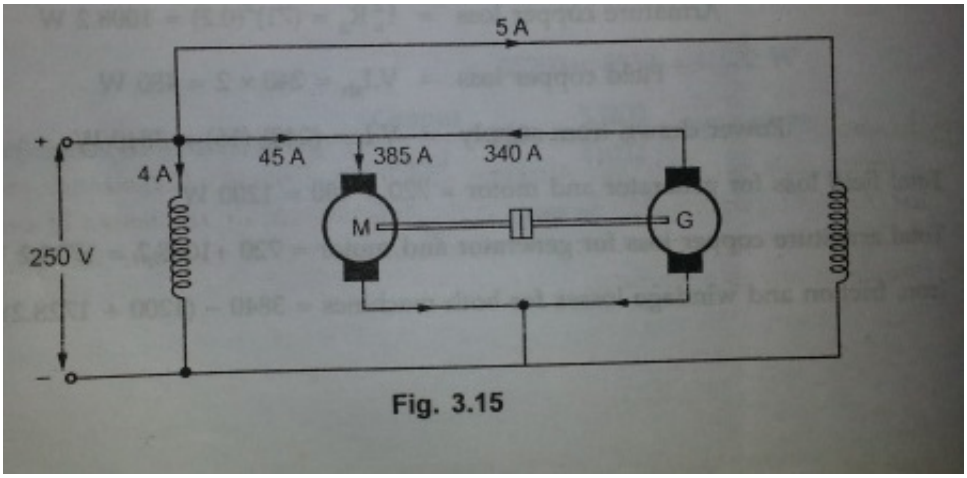


Figure 3.8: TO CALCULATE EFFICIENCY OF EACH OF THE 2 DC SHUNT MACHINES

```
    for generator
8  arm_cu_loss_m= R_a*I_a_m^2    //armature copper loss
    for motor
9  field_cu_loss_m= V*I_sh_m     //field copper loss for
     motor
10
11 arm_cu_loss_g= R_a*I_a_g^2  //armature copper loss
    for generator
12 field_cu_loss_g= V*I_sh_g    //field copper loss for
    motor
13
14 total_cu_loss = field_cu_loss_g + arm_cu_loss_g  +
    field_cu_loss_m + arm_cu_loss_m  //total copper
    loss for both machines
15 P_aux = V*I  //power taken from auxillary supply
16 stray_loss= P_aux - total_cu_loss
17 stray_loss_each = stray_loss/2  //stray loss for
    each machine
18
19 total_loss_g = stray_loss_each + arm_cu_loss_g +
    field_cu_loss_g  //total losses in generator
```

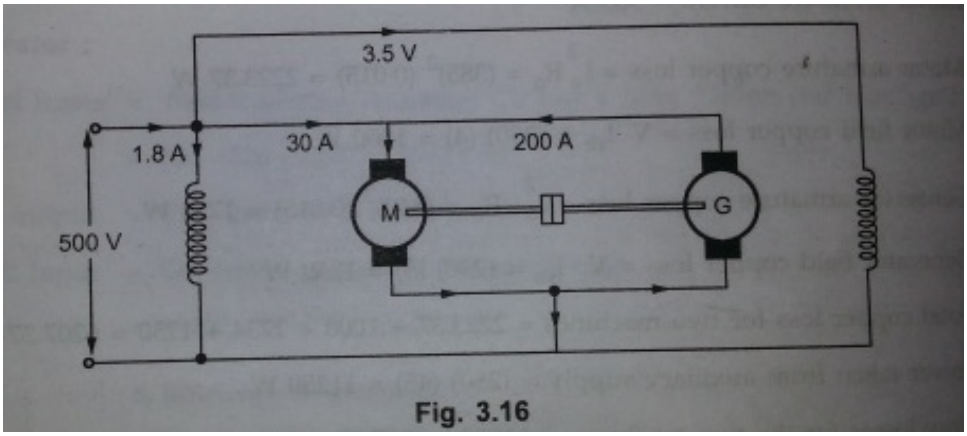


Figure 3.9: TO CALCULATE EFFICIENCY OF MACHINE ACTING AS GENERATOR

```
20 generator_output=V* I_a_g
21 eta_g = 100*(generator_output/(generator_output +
      total_loss_g))//generator efficiency
22 printf('Efficiency of generator is %.4f percent\n',
      eta_g)
23
24 total_loss_m = stray_loss_each + arm_cu_loss_m +
      field_cu_loss_m//total losses in motor
25 motor_input=V*(I_a_m+I_sh_m)
26 motor_output = motor_input - total_loss_m
27 eta_m = 100*(motor_output/motor_input)//motor
      efficiency
28 printf('Efficiency of motor is %.4f percent\n',eta_m
      )
```

Scilab code Exa 3.28 TO CALCULATE EFFICIENCY OF MACHINE ACTING AS GENERATOR

```
clc,clear
printf('Example 3.28\n\n')

V=500,P=1000*10^3,I=30
I_a_m = 200 + 30 , I_a_g =200 //armature current for motor and generator
I_sh_m = 1.8, I_sh_g =3.5 //field current for motor and generator
brush_drop=230
R_a=0.075  //armature resitance

arm_cu_loss_m = R_a*I_a_m^2 + 2*brush_drop //motor armature copper loss
field_cu_loss_m =V*I_sh_m  //  motor field copper loss

arm_cu_loss_g = R_a*I_a_g^2 + 2*brush_drop  // generator armature copper loss
field_cu_loss_g =V*I_sh_g  //field copper loss generator

total_cu_loss = field_cu_loss_g + arm_cu_loss_g  + field_cu_loss_m + arm_cu_loss_m  //total copper loss for both machines
P_aux = V*I  //power taken from auxillary supply
stray_loss= P_aux - total_cu_loss
stray_loss_each = stray_loss/2  //stray loss for each machine

total_loss_g = stray_loss_each + arm_cu_loss_g + field_cu_loss_g  //total loss in generator
generator_output=V* I_a_g
eta_g = 100*(generator_output/(generator_output + total_loss_g))//generator efficiency
printf('Efficiency of generator is %.0f percent\n', eta_g)
```

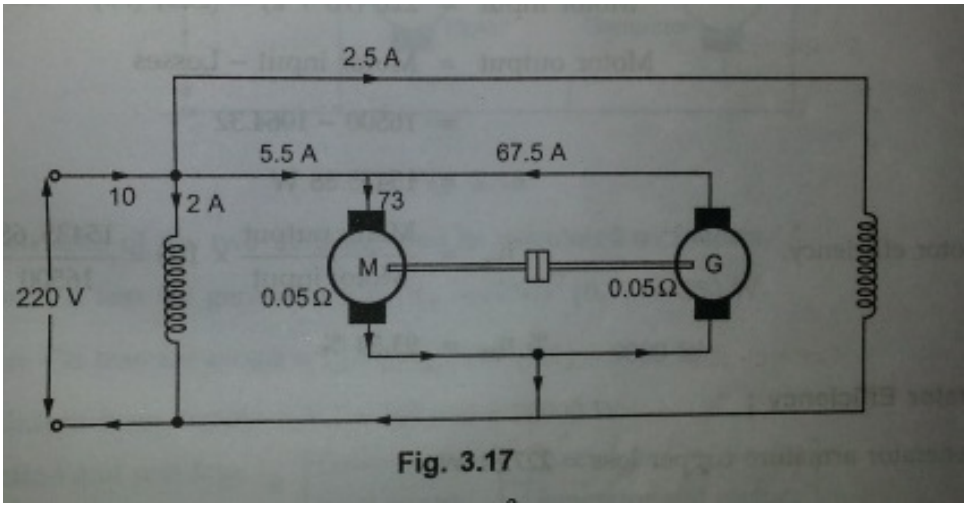


Figure 3.10: TO CALCULATE EFFICIENCY OF DC MACHINES

Scilab code Exa 3.29 TO CALCULATE EFFICIENCY OF DC MACHINES

```
1  clc,clear
2  printf('Example 3.29\n\n')
3
4  V=220, I=10
5  R_a=0.05 //artmature resistance
6  I_a_m= 73, I_sh_m = 2 //armature and field current
       for motor
7  I_a_g=67.5,I_sh_g=2.5//armature and field current
       for generator
8
9  arm_cu_loss_m = R_a*I_a_m^2 //motor armature copper
       loss
10 field_cu_loss_m =V*I_sh_m  //  motor field copper
       loss
```

```
11
12 arm_cu_loss_g = R_a*I_a_g^2 //generator armature
      copper loss
13 field_cu_loss_g =V*I_sh_g  //field copper loss
      generator
14
15 total_cu_loss = field_cu_loss_g + arm_cu_loss_g  +
      field_cu_loss_m + arm_cu_loss_m  //total copper
      loss for both machines
16 power_input = V*I
17 stray_loss= power_input - total_cu_loss
18 stray_loss_each = stray_loss/2  //stray loss for
      each machine
19
20 //motor efficiency
21 total_loss_m= field_cu_loss_m + stray_loss_each  +
      arm_cu_loss_m  //total motor losses
22 motor_input = V*(I_a_m + I_sh_m )
23 motor_output =motor_input - total_loss_m
24 eta_m = 100*(motor_output/motor_input)//motor
      efficiency
25 printf('Efficiency of motor is %.4f percent\n',eta_m
      )
26
27 //generator efficiency
28 total_loss_g= field_cu_loss_g + stray_loss_each  +
      arm_cu_loss_g  //total generator losses
29 generator_output =V*I_a_g
30 generator_input = generator_output +  total_loss_g
31 eta_g = 100*(generator_output/generator_input)//
      motor efficiency
32 printf('Efficiency of generator is %.4f percent\n',
      eta_g)
```

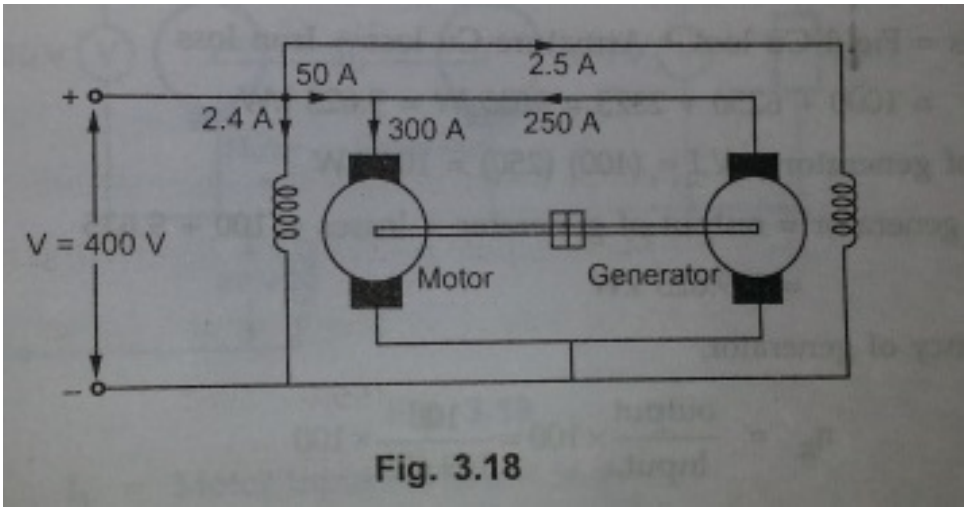


Figure 3.11: TO ESTIMATE EFFICIENCY OF 2 DC MACHINES

Scilab code Exa 3.30 TO ESTIMATE EFFICIENCY OF 2 DC MACHINES

```
clc,clear
printf('Example 3.30\n\n')

V=400, I=50
I_a_g=250 ,I_a_m =300  //armature current for
   generator and motor
I_sh_g=2.5 ,I_sh_m =2.4 //field current for
   generator and motor
R_a=0.1 //armature resistance

arm_cu_loss_g = R_a*I_a_g^2   //armature copper loss
    for generator
arm_cu_loss_m = R_a*I_a_m^2   //armature copper loss
    for motor
power_drawn=V*I
IFW_losses = power_drawn - (arm_cu_loss_g +
   arm_cu_loss_m)  //Iron , friction and windage
   losses
```

```
13 IFW_losses_each= IFW_losses /2  // Iron , friction
   and windage losses for each machine
14
15 //for motor
16 field_cu_loss_m= V*I_sh_m //field copper loss for
   motor
17 total_loss_m= field_cu_loss_m + IFW_losses_each  +
   arm_cu_loss_m
18 motor_input=V * I_a_m
19 motor_output= motor_input - total_loss_m
20 eta_m = 100*(motor_output/motor_input)    //motor
   efficiency
21 printf('Efficiency of motor is %.2f percent\n',eta_m
   )
22
23 //for generator
24 field_cu_loss_g= V*I_sh_g //field copper loss for
   generator
25 total_loss_g = field_cu_loss_g + arm_cu_loss_g +
   IFW_losses_each
26 generator_output=V*I_a_g
27 generator_input = generator_output + total_loss_g
28 eta_g = 100*(generator_output/generator_input)//
   generator efficiency
29 printf('Efficiency of generator is %.2f percent\n',
   eta_g)
```

Scilab code Exa 3.31 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR

```
1 clc,clear
2 printf('Example 3.31\n\n')
3
```

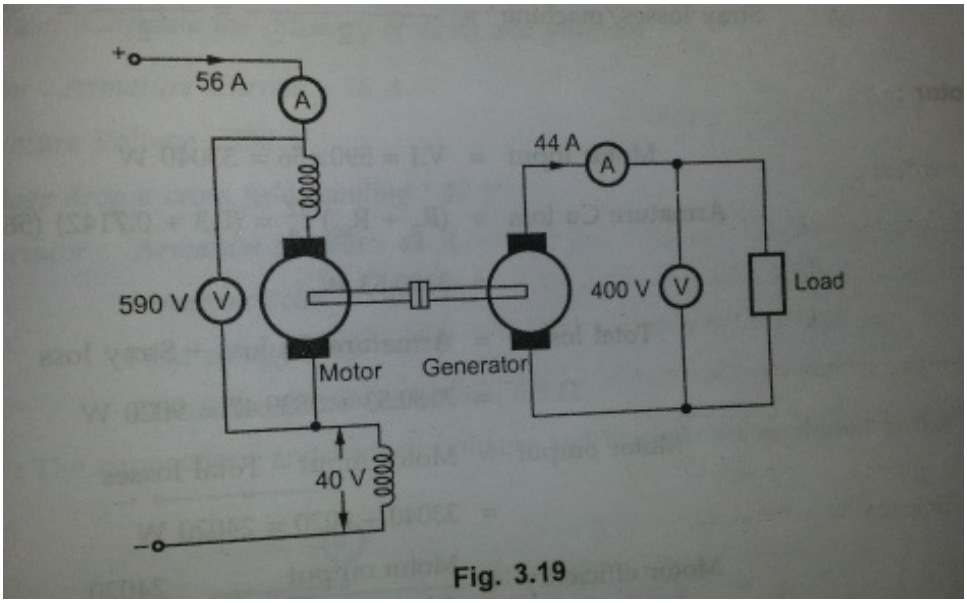


Figure 3.12: TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR

```
4  I_1=56  //motor input current
5  V=590   //voltage across armature
6  I_2=44  // load current
7  V_2=400 // voltage across generator
8  V_field = 40  //voltage drop across field winding
9  R_a=0.3, R_se=0.7142 //armature and series field
       resistance for each machine
10 total_input=(V+V_field)*I_1
11 output=V_2*I_2
12 total_loss_g_m= total_input - output   //total
       losses of 2 machines
13 R_se=V_field/I_1  //series field resistance for both
       windings
14 total_cu_loss = (R_a+ 2*R_se)*I_1^2  + R_a*I_2^2  //
       total copper loss
15 stray_loss= total_loss_g_m - total_cu_loss
16 stray_loss_each = stray_loss/2  //stray loss for
       each machine
```

```
17
18 // for motor
19 motor_input = V*I_1
20 arm_cu_loss_m =  (R_a+ R_se)*I_1^2    //armature
      coper losses of motor
21 total_loss_m= arm_cu_loss_m + stray_loss_each
22 motor_output = motor_input - total_loss_m
23 eta_m = 100*(motor_output/motor_input)//motor
      efficiency
24 printf('Efficiency of motor is %.4f percent\n',eta_m
      )
25
26 // for generator
27 arm_cu_loss_g = R_a*I_2^2  //armature coper losses
      of generator
28 series_field_cu_loss_g = V_field*I_1   //series
      field copper loss
29 total_loss_g= arm_cu_loss_g + series_field_cu_loss_g
       + stray_loss_each
30 generator_output=V_2*I_2
31 generator_input = generator_output +  total_loss_g
32 eta_g = 100*(generator_output/generator_input)//
      generator efficiency
33 printf('Efficiency of generator is %.4f percent\n',
      eta_g)
```

Scilab code Exa 3.32 TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR

```
1 clc,clear
2 printf('Example 3.32\n\n')
3
4 I_1=56  //motor input current
```

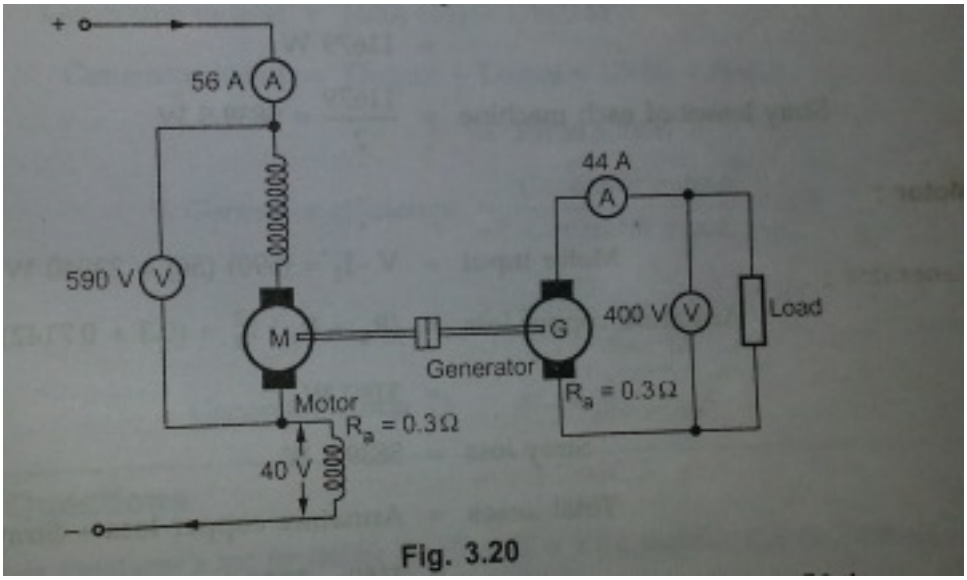


Figure 3.13: TO CALCULATE EFFICIENCY OF MOTOR AND GENERATOR

```
5  V=590    //voltage across armature
6  I_2=44   // load current
7  V_2=400 // voltage across generator
8  V_field = 40  //voltage drop across field winding
9  R_a=0.3, R_se=0.7142 //armature and series field
       resistane for each machine
10
11 total_input=(V+V_field)*I_1
12 output=V_2*I_2
13 total_loss_g_m= total_input - output    //total
       losses of 2 machines
14 R_se=V_field/I_1  //series field resistance for both
       windings
15 total_cu_loss = (R_a+ 2*R_se)*I_1^2  + R_a*I_2^2  //
       total copper loss
16 stray_loss= total_loss_g_m - total_cu_loss
17 stray_loss_each = stray_loss/2  //stray loss for
       each machine
18
```

```
// for motor
motor_input = V*I_1
arm_cu_loss_m =  (R_a+ R_se)*I_1^2  //armature coper losses of motor
total_loss_m= arm_cu_loss_m + stray_loss_each
motor_output = motor_input - total_loss_m
eta_m = 100*(motor_output/motor_input)//motor efficiency
printf('Efficiency of motor is %.4f percent\n',eta_m)

// for generator
arm_cu_loss_g = R_a*I_2^2  //armature coper losses of generator
series_field_cu_loss_g = V_field*I_1  //series field copper loss
total_loss_g= arm_cu_loss_g + series_field_cu_loss_g + stray_loss_each
generator_output=V_2*I_2
generator_input = generator_output +  total_loss_g
eta_g = 100*(generator_output/generator_input)// generator efficiency
printf('Efficiency of generator is %.4f percent\n', eta_g)
```

Chapter 4

Synchronous Machines Alternators

Scilab code Exa 4.1 TO DRAW THE DIAGRAM FOR FULL PITCH ARMATURE WINDING OF AN ALTERNATOR

```
1 clc,clear
2 printf('Example 4.1\n\n')
3
4 Pole=4
5 Slots=24
6 Phase=3 //number of phases
7 n=Slots/Pole //slots per pole
8 m=Slots/Pole/Phase //slots per pole per phase
9 beeta=180/n  //Slot angle
```

Scilab code Exa 4.2 TO CALCULATE DISTRIBUTION FACTOR OF THREE PHASE ALTERNATOR

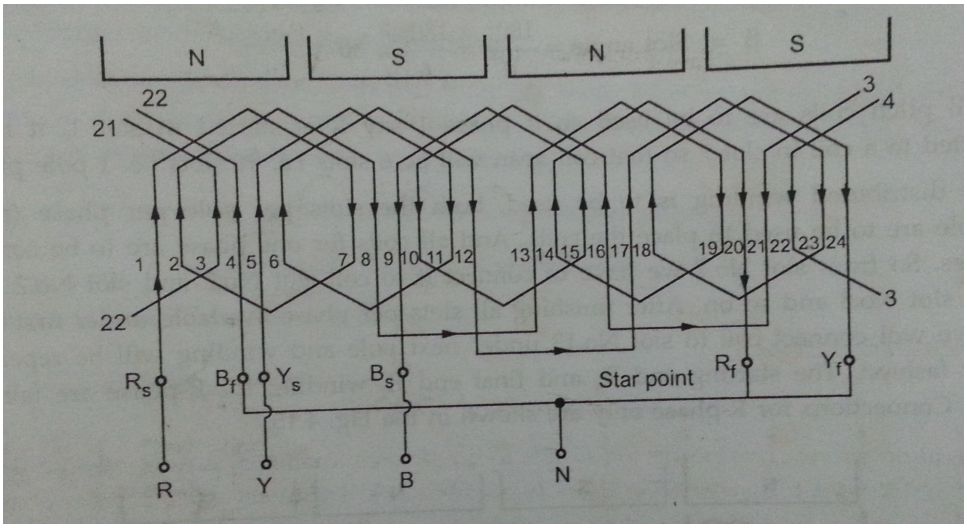


Figure 4.1: TO DRAW THE DIAGRAM FOR FULL PITCH ARMATURE WINDING OF AN ALTERNATOR

```
clc,clear
printf('Example 4.2\n\n')

Slots=120
Pole=8
Phase=3 //number of phases
n=Slots/Pole          //Slots per Pole
m=Slots/Pole/Phase    //Slots per Pole per Phase
beeta=180/n           //Slot angle in degree
K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
   Distribution Factor
printf('Distribution Factor:\nK_d=%.3f',K_d)
```

Scilab code Exa 4.3 TO CALCULATE COIL SPAN FACTOR OF ARMATURE WINDING

```
clc,clear
printf('Example 4.3\n\n')
```

```
3
4 Slots=36
5 Pole=4
6 Phase=3 //number of phases
7 n=Slots/Pole     //Slots per pole
8 beeta=180/n      //Slot angle in degrees
9
10 //coil is shorted by 1 slot i.e. by beeta degrees to
       full pitch distance
11 alpha=beeta      //angle of short pitch
12 K_c=cosd(alpha/2)  //Coil span Factor
13 printf('Coil Span Factor:\nK_c=%.4f',K_c)
```

Scilab code Exa 4.4 TO CALCULATE INDUCED EMF ACROSS THE TERMINALS

```
1 clc,clear
2 printf('Example 4.4\n\n')
3
4 N_s=250  //Synchronous speed in r.p.m
5 f=50     //Frequency of generated e.m.f in hertz
6 Slots=216
7 phi=30*10^-3 //flux per pole in weber
8
9 Pole=120*f/N_s
10 n=Slots/Pole   //Slots per Pole
11 m=n/3          //Slots per Pole per Phase
12 beeta=180/n     //Slot angle in degree
13
14 K_d=sind(m*beeta/2)/(m*sind(beeta/2))   //
      distribution factor
15 K_c=1 //Coil Span Factor for full pitch coils=1
16
17 Z=Slots*5   //Z is total no of conductors
18 Z_ph=Z/3    //Conductors Per Phase
```

```
19 T_ph=Z_ph/2 //Turns per phase
20 E_ph=4.44*K_c*K_d*f*phi*T_ph //induced emf
21 E_line=E_ph*sqrt(3)
22
23 printf('Induced e.m.f across the Terminals is %.2f V
   ',E_line)
```

Scilab code Exa 4.5 TO DETERMINE FREQUENCY OF INDUCED EMF and FLUX PER POLE

```
1 clc,clear
2 printf('Example 4.5\n\n')
3
4 Pole=16
5 N_s=375 //synchronous speed in rpm
6 Slots=144
7 E_line=2.657*10^3 //line value of emf across
   terminals
8 f=Pole*N_s/120 //frequency
9
10 K_c=1  //assuming full pitch winding,Coil span
   Factor=1
11 n=Slots/Pole //slots per pole
12 m=n/3 //slots per pole per phase
13
14 beeta=180/n
15 K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
   Distribution Fcator
16 conductors_per_slot=10
17 Z=Slots*conductors_per_slot  //total conductors
18
19 Z_ph=Z/3  //number of conductors per phase
20 T_ph=Z_ph/2 //no of turns per phase
21 E_ph=E_line/sqrt(3) //phase value of emf across
   terminals
```

```
22
23 phi=E_ph/(4.44*K_c*K_d*f*T_ph)      //E_ph=4.44*K_c*
      K_d*f*phi*T_ph
24 printf('Frequency of Induced e.m.f is %.0fHz\nFlux
      per Pole is %.0f mWb',f,phi*1000)
```

Scilab code Exa 4.6 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE ALTERNATOR

```
1  clc,clear
2  printf('Example 4.6\n\n')
3
4  d=0.25              //Diameter in metre
5  l=0.3               //Length in metre
6  Pole=4
7  A1=%pi*d*l/Pole     //Area of each fundamental pole
8  f=50 //frequency in hertz
9  B_m1=0.15 , B_m3=0.03, B_m5=0.02 //Amplitude of 1st,
       3rd and 5th harmonics
10 phi_1=(2/%pi)*B_m1*A1  //average value of
      fundamental flux per pole in weber
11
12 //PART A
13 E_c1=1.11*2*f*phi_1  //R.M.S value of fundamental
      frequency e.m.f generated in single conductor
14 Coil_span=(13/15)*180  //since winding coil span is
      13/15 of pole pitch
15 alpha=180-Coil_span
16
17 //Pitch factor for 1st, 3rd and 5th harmonic
18 K_c1=cosd(alpha/2)
19 K_c3=cosd(3*alpha/2)
20 K_c5=cosd(5*alpha/2)
21
22 //Using E_cx=E_c1 * (B_mx/B_m1)
```

```
E_c3=E_c1 * (B_m3/B_m1)
E_c5=E_c1 * (B_m5/B_m1)

E_t1=K_c1 * (2*E_c1)   //R.M.S Vaue of fundamental frequency EMF generated in 1 turn (in volts)
E_t3=K_c3 * 2*E_c3
E_t5=K_c5 * 2*E_c5
E_t=sqrt(E_t1^2 +E_t3^2 +E_t5^2)
V=10*E_t  //(number of turns per coil )* (Total e.m.f per turn)
printf('Voltage generated per coil is %.1f V',V)

// PART B
//E_1ph=4.44*K_c1*K_d1*phi_1*f*T_ph
T_ph=200   //T_ph=(60 coils  * 10 turns per coil)/3

Total_Conductors=1200 // 60 coils * 10 turns per coil * 2
Conductors_per_Slot=20 //2 conductors per turn * 10 turns per slot
Slots=Total_Conductors/Conductors_per_Slot

n=Slots/Pole
m=n/3
beeta=180/n   //Slot angle in degree
K_d1=sind(m*1*beeta/2) /(m*sind(1*beeta/2))
K_d3=sind(m*3*beeta/2) /(m*sind(3*beeta/2))
K_d5=sind(m*5*beeta/2) /(m*sind(5*beeta/2))

E_1ph=4.44 * K_c1 * K_d1*phi_1 * f * T_ph
// Using E_xph= E_1ph* (B_mx*K_cx*K_dx)/(B_m1*K_c1*K_d1)
E_3ph= E_1ph* (B_m3*K_c3*K_d3)/(B_m1*K_c1*K_d1)
E_5ph= E_1ph* (B_m5*K_c5*K_d5)/(B_m1*K_c1*K_d1)
E_ph=sqrt( E_1ph^2 + E_3ph^2 + E_5ph^2 )  //voltage generated per phase
printf('\nVoltage generated per phase is %.0f V', E_ph)
```

```
54
55  //PART c
56 E_line=sqrt(3) * sqrt( E_1ph^2 + E_5ph^2  ) //
      terminal voltage
57 printf('\nTerminal Voltage is %.1f V ',E_line)
```

Scilab code Exa 4.7 TO CALCULATE THE FLUX PER POLE OF 3 PHASE STAR CONNECTED ALTERNATOR

```
1  clc,clear
2  printf('Example 4.7\n\n')
3
4  Ns=250 //Synchronous speed in rpm
5  f=50
6  Slots=288
7  E_line=6600
8  Pole=120*f/Ns
9  n=Slots/Pole  //slots per pole
10 m=n/3 //slots per pole per phase
11 beeta=180/n //slot angle
12 conductors_per_slot=32   //16 conductors per coil-
      side  *2 coil-sides per slot
13
14 K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
      distribution factor
15 alpha=2*beeta// angle of short pitch
16 K_c=cosd(alpha/2)  //coil span factor
17 Z = Slots*conductors_per_slot  //total conductors
18 Z_ph=Z/3 //Conductors per phase
19 T_ph=Z_ph/2 //turns per phase
20
21 E_ph=E_line/sqrt(3)
22 phi=E_ph/(4.44*K_c*K_d*f*T_ph)           //Because
      E_ph=4.44 *K_c *K_d *phi *f *T_ph
23 printf('Flux per pole is %.0f mWb ',phi*1000)
```

Scilab code Exa 4.8 TO CALCULATE THE INDUCED EMF OF 1 PHASE ALTERNATOR

```
clc,clear
printf('Example 4.8\n\n')

Ns=1500  //synchronous speed in rpm
Pole=4
Slots=24
conductor_per_slot=8
phi=0.05 //flux per pole in weber
f=Pole*Ns/120 //frequenccy
n=Slots/Pole //slots per pole
m=n  // as number of phases is 1
beeta=180/n  //slot angle

K_d=sind(m*beeta/2) /(m*sind(beeta/2))  //
   distribution factor

//Full pitch= n =6 slots
//(1/6)th of full pitch =1slot
//angle of short pitch = 1 slot angle
alpha=beeta
K_c=cosd(alpha/2)  //coil span factor

Z=conductor_per_slot*Slots //total conductors
Z_ph=Z // as number of phases is 1
T_ph=Z_ph/2 //turns per phase
E_ph=4.44*K_c*K_d* phi *f *T_ph  //induced emf

printf('Induced e.m.f is %.1f V ',E_ph)
```

Scilab code Exa 4.9 TO DETERMINE INDUCED EMF BETWEEN THE LINES OF 3 PHASE STAR CONNECTED ALTERNATORS

```
clc,clear
printf('Example 4.9\n\n')

Pole=48
n=9 //slots per pole
phi=51.75*10^-3 //flux per pole in weber
Ns=125
f=Ns*Pole/120 //frequency
K_c=1 //due to full pitch winding
m=n/3 //slots per pole per phase
beeta=180/n  //slot angle

K_d=sind(m*beeta/2) /(m*sind(beeta/2))  //
   distribution factor
conductor_per_slot=4*2 //Each slot has 2 coil sides
   and each coil side has 4 conductors
Slots=n*Pole
Z=conductor_per_slot*Slots   //total number of
   conductors
Z_ph=Z/3 //conductors per phase
T_ph=Z_ph/2 //turns per phase
E_ph=4.44 *K_c *K_d *phi *f *T_ph  //induced emf

E_line=(sqrt(3))*E_ph  //due to star connection
printf('Induced e.m.f is %.0f kV ',E_line/1000)
```

Scilab code Exa 4.10 TO DETERMINE CERTAIN QUANTITIES RELATED TO 12 POLE 3 PHASE STAR CONNECTED ALTERNATOR

```
clc,clear
printf('Example 4.10\n\n')

```

```
4  Slots=180
5  Pole=12
6  Ns=600 //Synchronous speen in rpm
7  f=Pole*Ns/120 //frequency
8  phi=0.05 //flux per pole in weber
9
10 //Part(i)
11 //Average EMF in a conductor=2*f*phi
12 rms_value_1=1.11*2*f*phi //rms value of emf in a
      conductor
13 printf('(i)r.m.s value of e.m.f in a conductor is %
      .2f V ',rms_value_1)
14
15 //part(ii)
16 //Average EMF in a turn=4*f*phi
17 rms_value_2=1.11*4*f*phi//r.m.s value of e.m.f in a
      turn
18 printf('\n(ii)r.m.s value of e.m.f in a turn is %.2f
       V ',rms_value_2)
19
20 //part(iii)
21 conductors_per_coilside=10/2
22 rms_value_3=rms_value_2*conductors_per_coilside  //r
      .m.s value of e.m.f in a coil
23 printf('\n(iii)r.m.s value of e.m.f in a coil is %.1
      f V ',rms_value_3)
24
25 //part(iv)
26 conductors_per_slot=10
27 Z=conductors_per_slot * Slots  //total number of
      conductors
28 Z_ph=Z/3    //conductors per phase
29 T_ph=Z_ph/2 //turns per phase
30 n=Slots/Pole  //slots per pole
31 m=n/3        //slots per pole per phase
32 beeta=180/n   //slot angle
33
34 K_d=sind(m*beeta/2) /(m*sind(beeta/2)),K_c=1  //
```

```
    distribution & coil-span factor
35 E_ph=rms_value_2*T_ph*K_d*K_c    //induced emf
36 printf('\n(iv)per phase induced e.m.f is %.1f V ',
    E_ph)
```

Scilab code Exa 4.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED ALTERNATORS

```
1  clc,clear
2  printf('Example 4.11\n\n')
3
4  Pole=8
5  f=50 //frequency
6  phi=60*10^-3 //flux per pole in weber
7  Slots=96
8  n=Slots/Pole //slots per pole
9  beeta = 180/n //slot angle
10 m=n/3 //slots per pole per phase
11
12 coil_pitch=10*beeta //10 slots
13 alpha=180-coil_pitch
14 K_c=cosd(alpha/2)    //coi;-span factor
15 K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
    distribution factor
16
17 conductors_per_slot=4
18 Z=Slots*conductors_per_slot //total conductors
19 Total_turns=Z/2
20 T_ph=Total_turns/3  //turns per phase
21
22 //part (i)
23 E_ph= 4.44 *K_c *K_d *phi *f *T_ph
24 printf('\The phase voltage is %.2f V ',E_ph)
25
26 //part(ii)
```

```
27 E_line=E_ph*sqrt(3)
28 printf('\nThe Line Voltage is %.2f V ',E_line)
29
30  //part(iii)
31 I_ph=650
32 I_l=I_ph  // Star Connection
33 kVA_rating=sqrt(3)*E_line*I_l
34 printf('\nkVA rating is %.1f kVA ',kVA_rating/1000)
```

Scilab code Exa 4.12 TO DETERMINE INDUCED EMF IN 3 PHASE ALTERNATOR

```
1  clc,clear
2  printf('Example 4.12\n\n')
3
4  Ns=600 //synchronous speed in rpm
5  Pole=10
6  l=30/100      //divided by 100 for centimetre-metre
      conversion
7  Pole_pitch=35/100  //numerically equal to pi*d/Pole
8  Phase=3
9  conductors_per_slot=8
10 A1=Pole_pitch*l //Area of each fundamental pole
11 m=3   //Slot per Pole per Phase
12 n=Phase*m //slots per pole
13 beeta=180/n  //slot angle
14
15 B_m1=1,B_m3=0.3,B_m5=0.2  //amplitude of 1st, 3rd
      and 5th harmonic
16 phi_1=(2/%pi)*A1*B_m1  //average value of
      fundamental flux per pole
17 f=Ns*Pole/120 //frequency
18
19 Coil_span=(8/9)*180
20 alpha=180-Coil_span
```

```
21 //pitch factor for 1st, 3rd and 5th harmonic
22 K_c1=cosd(alpha/2)
23 K_c3=cosd(3*alpha/2)
24 K_c5=cosd(5*alpha/2)
25
26 // using K_dx=sin(m*x*beeta*(%pi/180)/2) /(m*sin(x*
   beeta*(%pi/180)/2))
27 //distribution factor for 1st, 3rd and 5th harmonic
28 K_d1=sind(m*1*beeta/2) /(m*sind(1*beeta/2))
29 K_d3=sind(m*3*beeta/2) /(m*sind(3*beeta/2))
30 K_d5=sind(m*5*beeta/2) /(m*sind(5*beeta/2))
31
32 Slots=n*Pole
33 Total_conductors=conductors_per_slot * Slots
34 Total_turns=Total_conductors/2
35 T_ph=Total_turns/3  //turns per phase
36
37 //EMF of 1st , 3rd and 5th harmonic
38 E_1ph=4.44 * K_c1 * K_d1*phi_1 * f * T_ph
39 E_3ph= E_1ph* (B_m3*K_c3*K_d3)/(B_m1*K_c1*K_d1)
40 E_5ph= E_1ph* (B_m5*K_c5*K_d5)/(B_m1*K_c1*K_d1)
41
42 // Using E_xph= E_1ph* (B_mx*K_cx*K_dx)/(B_m1*K_c1*
   K_d1)
43 E_ph=sqrt( E_1ph^2 + E_3ph^2 + E_5ph^2 )
44 printf('Phase value of induced e.m.f is %.2f V ',
   E_ph)
45 E_line=sqrt(3) * sqrt( E_1ph^2 + E_5ph^2  )//no 3rd
   harmonic appears in line value
46 printf('\nline value of induced e.m.f is %.2f V ',
   E_line)
47
48 printf('\n\nAnswer mismatches due to approximation')
```

Scilab code Exa 4.13 TO CALCULATE FREQUENCY AND LINE VOLTAGE OF 3PHASE ALTERNATOR

```
clc,clear
printf('Example 4.13\n\n')

Pole=16
phi=0.03 //flux per pole
Ns=375 //synchronous speed in rpm
f=Ns*Pole/120 //frequency
printf('frequency is %.0f Hz ',f)
Slots=144
n=Slots/Pole  //slots per pole
m=n/3 //slots per pole per phase
beeta=180/n  //slot angle
K_c=1 //assuming Full-Pitch coil
Conductors_per_slot=10
K_d=sind(m*beeta/2) /(m*sind(beeta/2))  //
   distribution factor

Total_conductors=Slots*Conductors_per_slot
Total_turns=Total_conductors/2
T_ph=Total_turns/3  //turns per phase
E_ph=4.44* K_c* K_d*phi* f* T_ph
E_line=E_ph*sqrt(3)
printf('\nline voltage is %.2f V ',E_line)
```

Scilab code Exa 4.14 TO DETERMINE kVA RATING OF A SYNCHRONOUS GENERATOR

```
clc,clear
printf('Example 4.14\n\n')

Ns=250 //Speed in rpm
f=50 //frequency
```

```
I_l=100
Slots=216
Conductors_per_slot=5
Pole=120*f/Ns
phi=30*10^-3//flux per pole in weber
Z=Slots*Conductors_per_slot  //Total Conductors
Z_ph=Z/3 //conductors per phase
T_ph=Z_ph/2 //turns per phase
n=Slots/Pole //slots per pole
m=n/3 //slots per pole per phase
beeta=180/n //Slot angle

K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
   distribution factor

e_av=2*f*phi     //Average Value of EMF in each
   conductor
E_c=1.11*(2*f*phi)  //RMS value of EMF in each
   conductor
E=2*E_c*K_d //RMS value of EMF in each turn
E_ph=T_ph*E //RMS value of EMF in each phase
E_line= E_ph*sqrt(3)  //As Star Connected Alternator
printf('RMS value of EMF in each phase = %.3f V\n',
   E_ph)
printf('RMS value of EMF line value  = %.3f V\n',
   E_line)
kVA_rating=sqrt(3)*E_line*I_l
printf('\nkVA rating is %.3f kVA ',kVA_rating/1000)
```

Scilab code Exa 4.15 TO DETERMINE THE NUMBER OF ARMATURE CONDUCTORS REQUIRED TO GIVE A LINE VOLTAGE OF 11kV

```
clc,clear
printf('Example 4.15\n\n')

```

```
Pole=10
Slots=90
E_l=11000
f=50
phi=0.15 //flux per pole in weber
n=Slots/Pole //slots per pole
m=n/3 //slots per pole per phase
beeta=180/n //slot angle

K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
   distribution factor
K_c=1 //coil span factor

E_ph=E_l/sqrt(3)
T_ph=floor( E_ph/(4.44*K_c*K_d*phi*f)  )
//T_ph should necessarily be an integer

Z_ph=(T_ph)*2
printf('Required number of armature conductors is %d
   ',Z_ph)
```

Scilab code Exa 4.16 TO DETERMINE RMS VALUE OF PHASE AND LINE VOLTAGE

```
clc,clear
printf('Example 4.16\n\n')

Pole=10
Ns=600 //speen in rpm
conductor_per_slot=8
n=12 //slots per pole
Slots=Pole*n
m=n/3 //slots per pole per phase
beeta=180/n //slot angle
alpha=2*beeta  //short by 2 slots
```

```
12
13 //flux per pole corresponding to 1st,3rd and 5th
      harmonic
14 phi_1=100*10^-3
15 phi_3=(33/100)*phi_1
16 phi_5=(20/100)*phi_1
17
18 //coil span factor corresponding to 1st,3rd and 5th
      harmonic
19 K_c1=cosd( alpha/2)
20 K_c3=cosd( 3*alpha/2)
21 K_c5=cosd( 5*alpha/2)
22
23 // using K_dx=sin(m*x*beeta /2) /(m*sin(x*beeta /2))
24 //distribution factor corresponding to 1st,3rd and 5
      th harmonic
25 K_d1=sind(m*1*beeta/2) /(m*sind(1*beeta /2))
26 K_d3=sind(m*3*beeta/2) /(m*sind(3*beeta /2))
27 K_d5=sind(m*5*beeta/2) /(m*sind(5*beeta /2))
28
29 Z=conductor_per_slot*n*Pole    //Total Conductors
30 Zph=Z/3 //conductors per phase
31 T_ph=Zph/2 //turns per phase
32
33 f=Ns*Pole/120
34 E_1ph=4.44*K_c1*K_d1*phi_1*f*T_ph
35 E_3ph=4.44*K_c3*K_d3*phi_3*f*T_ph
36 E_5ph=4.44*K_c5*K_d5*phi_5*f*T_ph
37
38 E_ph=sqrt( E_1ph^2 +  E_3ph^2 + E_5ph^2     )
39 printf('Phase value of induced e.m.f is %.0f V ',
      E_ph)
40 E_line=sqrt(3)*sqrt( E_1ph^2 +  E_5ph^2     )  //In
      a line value,3rd harmonic doesnt appear
41 printf('\nline value of induced e.m.f is %.0f V ',
      E_line)
```

Scilab code Exa 4.17 TO DETERMINE RESULTANT PHASE VOLTAGE AND LINE VOLTAGE

```
clc,clear
printf('Example 4.17\n\n')

Pole=6
Ns=1000 //speed in rpm
d=28/100  //Divided by 100 to convert from
    centimeters to metres
l=23/100  //Divided by 100 to convert from
    centimeters to metres
m=4 //slots per pole per phase
B_m1=0.87  //amplitude of 1st harmonic component of
    flux density
B_m3=0.24  //amplitude of 3rd harmonic component of
    flux density
Conductors_per_slot=8
f=Ns*Pole/120 //frequency
A1=%pi*d*l/Pole //area of each fundamental pole
phi_1=(2/%pi)*A1*B_m1 //flux per pole in weber
n=m*3 //slots per pole
beeta=180/n //slot angle
alpha=beeta  //because of 1 slot short
K_c1=cosd(alpha/2) //coil span factor corresponding
    to 1st harmonic
K_c3=cosd(3*alpha/2)//coil span factor corresponding
     to 3rd harmonic
// using K_dx=sin(m*x*beeta*(%pi/180)/2) /(m*sin(x*
    beeta*(%pi/180)/2))
K_d1=sind(m*1*beeta/2) /(m*sind(1*beeta/2)) //
    distribution factor corresponding to 1st harmonic
K_d3=sind(m*3*beeta/2) /(m*sind(3*beeta/2)) //
    distribution factor corresponding to 3rd harmonic
```

```
23
24 Slots=n*Pole
25 Z=Slots*Conductors_per_slot //total number of
     conductors
26 Z_ph=Z/3 //conductors per phase
27 T_ph=Z_ph/2 //turns per phase
28
29 E_1ph=4.44*K_c1*K_d1*phi_1*f*T_ph
30 E_3ph=E_1ph* (B_m3*K_c3*K_d3)/(B_m1*K_c1*K_d1)
     //using E_xph=E_1ph* (B_mx*K_cx*K_dx)/(B_m1*K_c1*
     K_d1)
31 E_ph=sqrt( E_1ph^2 +  E_3ph^2     )
32 printf('r.m.s value of resultant voltage is %.1f V',
     E_ph)
33 E_line=sqrt(3)*E_1ph   //For line Value, 3rd
     harmonic does not appear
34 printf('\nline voltage is %.3f V',E_line)
```

Scilab code Exa 4.18 TO DETERMINE THE RATINGS WHEN DELTA CONNECTED ALTERNATOR IS RECONNECTED IN STAR

```
1 clc,clear
2 printf('Example 4.18\n\n')
3
4 V_L=125
5 V_ph=V_L
6 VA=600*10^3
7 I_L=VA/(sqrt(3)*V_L)      // Because VA=sqrt(3)* V_L
     * I_L
8 I_ph=I_L/(sqrt(3))
9
10 //After Reconnection
11 V_ph=125
12 V_L=V_ph*sqrt(3)
13 printf('New rating in volts is %.3f V',V_L)
```

```
14 //Winding Impedances remain the same
15 I_ph=1600
16 I_L=I_ph
17
18 printf('\nNew rating in amperes is %.0f A',I_L)
19 kVA=sqrt(3)*V_L*I_L*(10^-3)
20 printf('\nNew rating in kVA is %.0f kVA',kVA)
```

Scilab code Exa 4.19 TO CALCULATE GENERATED EMF OF 3 PHASE STAR CONNECTED ALTERNATOR

```
1  clc,clear
2  printf('Example 4.19\n\n')
3
4  Pole=4
5  f=50 //frequency
6  phi=0.12 //flux per pole in weber
7  m=4  // slot per pole per phase
8  conductor_per_slot=4
9  coilspan=150
10 Ns=120*f/Pole //synchronous speed in rpm
11 n=m*3  //Slots per pole
12 beeta=180/n //slot angle
13 K_d=sind(m*beeta/2) /(m*sind(beeta/2)) //
     distribution factor
14 alpha=180-coilspan //angle of short pitch
15 K_c=cos((%pi/180)*alpha/2) //coil span factor
16 Z=m*(n*Pole) // Also equal to (conductors/slots)*
     slots
17 Z_ph=Z/3 //conductors per phase
18 T_ph=Z_ph/2 //turns per phase
19 E_ph=4.44*K_c*K_d*phi*f*T_ph
20 E_line=sqrt(3)*E_ph
21 printf('e.m.f generated is %.2f V(phase),%.2f V(line
     )',E_ph,E_line)
```

Chapter 5

Methods for Calculating Regulation of Alternator

Scilab code Exa 5.1 TO DETERMINE EMF AND REGULATION AT A CERTAIN LOAD

```
1  clc,clear
2  printf('Example 5.1\n\n')
3
4  P=1000*10^3 //load power
5  phi=acosd(0.8) //power factor lagging angle
6  V_L=11*10^3 //rated terminal voltae
7  R_a=0.4 //armature resistance per phase
8  X_s=3//synchronous reactance per phase
9
10 I_L=P/(sqrt(3)*V_L*cosd(phi))
11 I_aph=I_L //for star connected load
12 I_a=I_L//current through armature
13 V_ph=V_L/sqrt(3) //rated terminal volatge phase
      value
14
15 E_ph= sqrt(  (V_ph*cosd(phi)+I_a*R_a)^2+(V_ph*sind(
      phi)+I_a*X_s)^2   )  //emf generated phase value
16 E_line=E_ph*sqrt(3) //line value of emf generated
```

```
17 regulation=100*(E_ph-V_ph)/V_ph  //pecentage
     regulation
18 printf('Line value of e.m.f generated is %.2f kV\
     nRegulation is %.3f percent',E_line*10^-3,
     regulation)
```

Scilab code Exa 5.2 TO DETERMINE PERCENTAGE REGULATION AT FULL LOAD LEADING AND LAGGING PF

```
1  clc,clear
2  printf('Example 5.2\n\n')
3
4  VA=1200*10^3
5  V_L=6600
6  R_a=0.25 //armature resistance per phase
7  X_s=5//synchronous reactance per phase
8
9  I_L=VA/(sqrt(3)*V_L)
10 I_aph=I_L //for star connected load
11 I_a=I_L
12 V_ph=V_L/sqrt(3)
13
14 //Part(i)
15 phi1=acos(0.8)//and lagging
16 E_ph1= sqrt(  (V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(
     phi1)+I_a*X_s)^2   )
17 regulation=100*(E_ph1-V_ph)/V_ph  //percentage
     regulation
18 printf('(i)Regulation at 0.8 lagging pf is %.2f
     percent',regulation)
19 //Part(ii)
20 phi2=acos(0.8)//and leading
21 E_ph2= sqrt(  (V_ph*cos(phi2)+I_a*R_a)^2+(V_ph*sin(
     phi2)-I_a*X_s)^2   )
22 regulation2=100*(E_ph2-V_ph)/V_ph //percentage
```

```
    regulation
23 printf('\n(ii)Regulation at 0.8 leading pf is %.2f
    percent',regulation2)
```

Scilab code Exa 5.3 TO DETERMINE PERCENTAGE REGULATION ON FULL LOAD

```
1  clc,clear
2  printf('Example 5.3\n\n')
3
4  //full load
5  V_L_FL=1100
6  V_ph_FL=V_L_FL/sqrt(3)
7
8  //no load
9  V_L_NL=1266
10 E_line=V_L_NL
11 E_ph=E_line/sqrt(3)
12 regulation=100*(E_ph-V_ph_FL)/V_ph_FL
13
14 printf('Regulation at full load is %.2f percent',
    regulation)
```

Scilab code Exa 5.4 TO CALCULATE FULL LOAD REGULATION AT A LAGGING POWER FACTOR

```
1  clc,clear
2  printf('Example 5.4\n\n')
3
4  V_L=866
5  VA=100*10^3
```

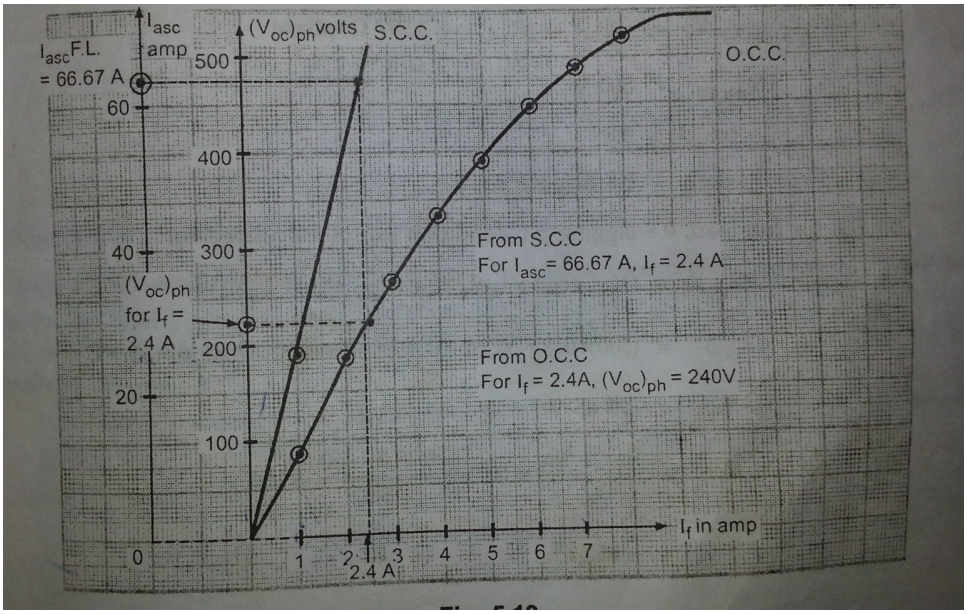


Figure 5.1: TO CALCULATE FULL LOAD REGULATION AT A LAGGING POWER FACTOR

```
6  I_L=VA/(sqrt(3)*V_L)     //because VA=sqrt(3)*V_L*I_L
7  I_aph=I_L//full load and star connected alternator
8  V_ph=V_L/sqrt(3)
9
10 //Graph is plotted and V_oc_ph and I_asc_Ph is
       obtained for
11 //SCC for I_asc=66.67 A, I_f=2.4 A
12 //OCC for I_f=2.4 A, V_oc_ph=240 V
13
14 //for measruemnt of impedance
15 V_oc_ph=240 //for I_f=2.4..From o.c.c graph
16 I_asc_ph=66.67 //for I_f=2.4...From s.c.c graph
17 Z_s=V_oc_ph/I_asc_ph
18 R_a=0.15
19 X_s=sqrt( Z_s^2-R_a^2 )
20
21 V_ph_FL=500
22 phi=acos(0.8)  //lagging pf
```

```
23 E_ph=sqrt((V_ph_FL*cos(phi)+I_aph*R_a)^2+(V_ph_FL*
     sin(phi)+I_aph*X_s)^2)
24 regulation=100*(E_ph-V_ph)/V_ph
25
26 printf('Full-load regulation at 0.8 lagging pf is %
     .2f percent ',regulation )
```

Scilab code Exa 5.5 TO FIND PERCENTAGE REGULATION AT CERTAIN LEADING AND LAGGING POWER FACTORS

```
1  clc,clear
2  printf('Example 5.5\n\n')
3
4  V_OC_line=230,I_asc=12.5  // when I_f=0.38
5  V_OC_ph=V_OC_line/sqrt(3)
6  Z_s=V_OC_ph/I_asc
7
8  R_a=1.8/2  //1.8 is between terminals..0.9 is per
     phase
9  X_s=sqrt(Z_s^2-R_a^2)
10
11 I_a=10// when regulation is needed
12 V_L=230
13 V_ph=V_L/sqrt(3)
14
15 //Part(i)
16 phi1=acos(0.8) //and lagging
17 E_ph1=sqrt((V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(phi1
     )+I_a*X_s)^2)
18 regulation1=100*(E_ph1-V_ph)/V_ph
19 printf('Regulation for 10 A at 0.8 lagging pf is %.2
     f percent\n',regulation1)
20 //Part(ii)
21 phi2=acos(0.8) //and leading
22 E_ph2=sqrt((V_ph*cos(phi2)+I_a*R_a)^2+(V_ph*sin(phi2
```

```
    )-I_a*X_s)^2)
23 regulation2=100*(E_ph2-V_ph)/V_ph
24 printf('Regulation for 10 A at 0.8 leading pf is %.2
    f percent\n',regulation2)
```

Scilab code Exa 5.6 TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD

```
1  clc,clear
2  printf('Example 5.6\n\n')
3
4  phi=acos(0.8)
5  VA=1000*10^3
6  V_L=1905
7  V_ph=V_L/sqrt(3)
8  R_a=0.2//Armature reactance per phase
9
10 //Part(i)
11 //Ampere-turn method
12 I_L=VA/(sqrt(3)*V_L)
13 I_aph=I_L
14 V_dash=V_ph+I_aph*R_a*cos(phi)//V_dash is a dummy
    quantity and has no significance..it's used only
    for mapping correcponding current
15 F_o=32 //F_o corresponds to voltage V_dash=1148.5
    from O.C.C graph
16 F_AR=27.5 //Field current required to circulate full
    -load short circuit current of 303.07A.From SCC
    F_AR=27.5
17 F_R = sqrt(    F_o^2 + F_AR^2-2*F_o*F_AR*cos(phi+%pi
    /2) )//Using Cosine rule
18
19 // for F_R=53.25, E_ph=1490 V from O.C.C
20 E_ph=1490
21 regulation1=100*(E_ph-V_ph)/V_ph
```

```
22 printf('Regulation on full-load by ampere-turn
      method is %.2f percent',regulation1)
23
24 //Part (ii)
25 //Synchronous Impedance method
26
27 I_sc=I_L
28 I_aph2=I_sc
29 I_f=27.5
30
31 V_OC_ph=1060 //corresponding to I-f=27.5 in the
      graph
32 Z_s=V_OC_ph/I_aph2
33 X_s=sqrt(Z_s^2-R_a^2)
34
35 E_ph2= sqrt( (V_ph*cos(phi)+I_aph2*R_a)^2+(V_ph*sin(
      phi)+I_aph2*X_s)^2 )    //from phasor diagram
36 regulation2=100*(E_ph2-V_ph)/V_ph
37 printf('\nRegulation on full-load by synchronous
      impedance method is %.2f percent',regulation2)
```

Scilab code Exa 5.7 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS

```
1 clc,clear
2 printf('Example 5.7\n\n')
3
4 //case(i)
5 V_L=440
6 V_ph=V_L/sqrt(3)
7 phi=acos(0.8)
```

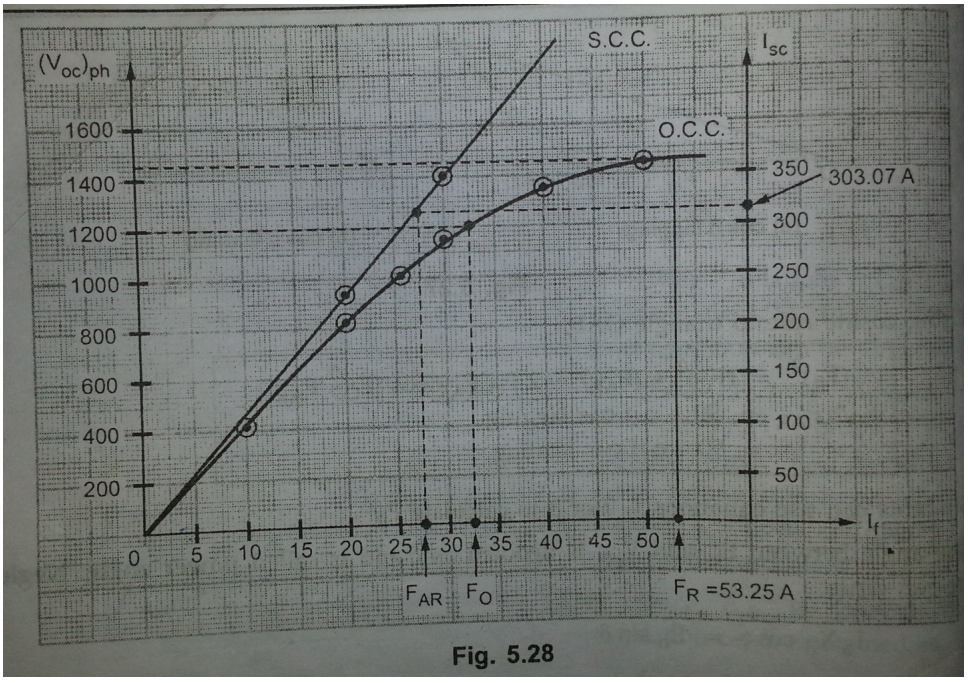
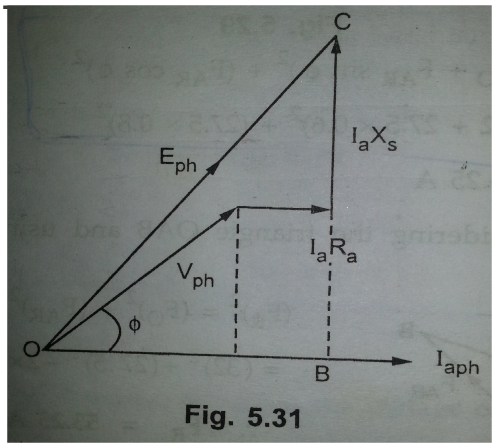
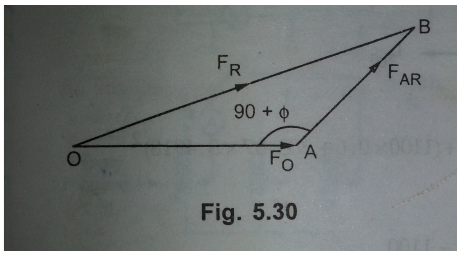


Figure 5.2: TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD



Ampere-turn Method



Synchronous Impedance method

Figure 5.3: TO FIND THE REGULATION ON FULL LOAD BY AMPERE TRUN METHOD AND SYNCHRONOUS IMPEDANCE METHOD

```
8
9  //armature resistance drop from the graph
10 //RS=1.1 cm and scale =50 V/cm
11 arm_leak_resis= 1.1*50    //armature leakage
       resistance
12
13 OB=V_ph*cos(phi)
14 AB=V_ph*sin(phi) + arm_leak_resis
15 E_1ph= sqrt( OB^2+AB^2   )
16
17 F_f1=6.1  //corresponding value from OCC
18 F_AR=3.1*1
19
20 F_R= sqrt(   F_f1^2 + F_AR^2 -2*F_f1*F_AR*cosd(90+
       acosd(0.8))           )
21 E_ph=328 //voltage corresponding to F_R=8.33 A from
       OCC graph
22 regulation1= 100*(E_ph - V_ph)/V_ph
23 printf('(i)Regulation for 0.8 pf lagging is %.2f
       percent \n',regulation1)
24
25 //case(ii)
26
27 OC=V_ph*cos(phi)
28 BC=V_ph*sin(phi) - arm_leak_resis
29 E_1ph= sqrt( OC^2+BC^2   )
30
31 F_f1=6.1  //corresponding value from OCC
32 F_R= sqrt(   F_f1^2 + F_AR^2 -2*F_f1*F_AR*cosd(90-
       acosd(0.8))           )
33 E_ph=90 //volatge corresponding to F_R=3.34 A from
       OCC graph
34 regulation2= 100*(E_ph - V_ph)/V_ph
35 printf('(ii)Regulation for 0.8 pf leading is %.2f
       percent \n',regulation2)
36 printf('\nThe answer in part (ii) doesnt match with
       textbook because of calculation mistake done in
       last step in the textbook')
```

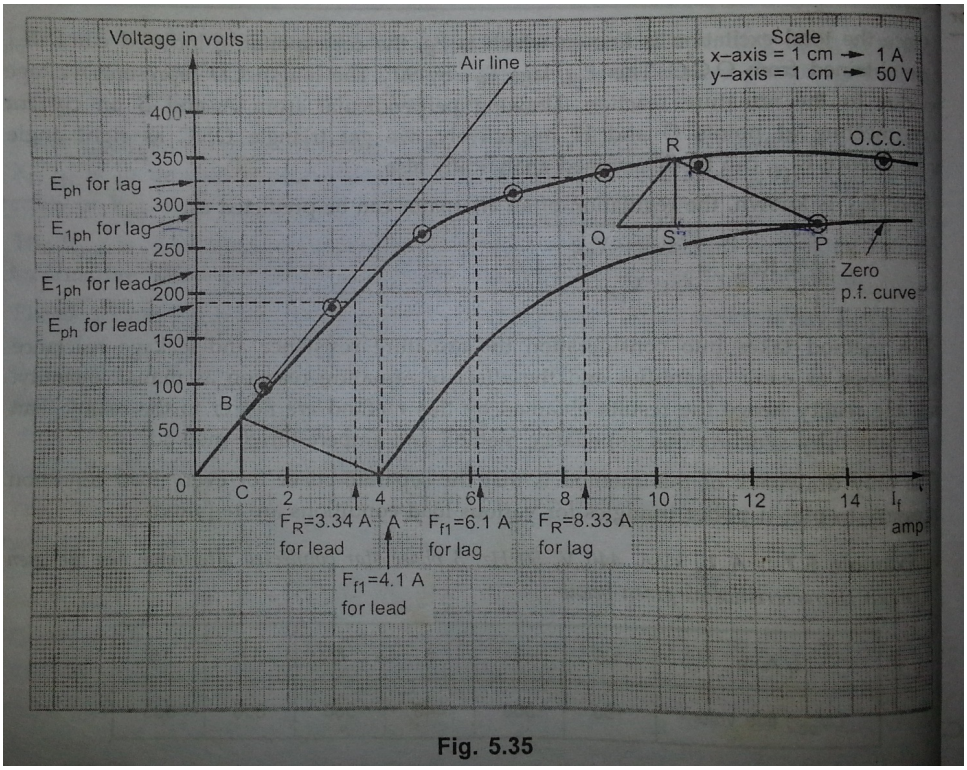


Figure 5.4: TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS

Scilab code Exa 5.8 TO DETERMINE PERCENTAGE REGULATION AT CERTAIN LAGGING POWER FACTOR

```
1 clc,clear
2 printf('Example 5.8\n\n')
3
```

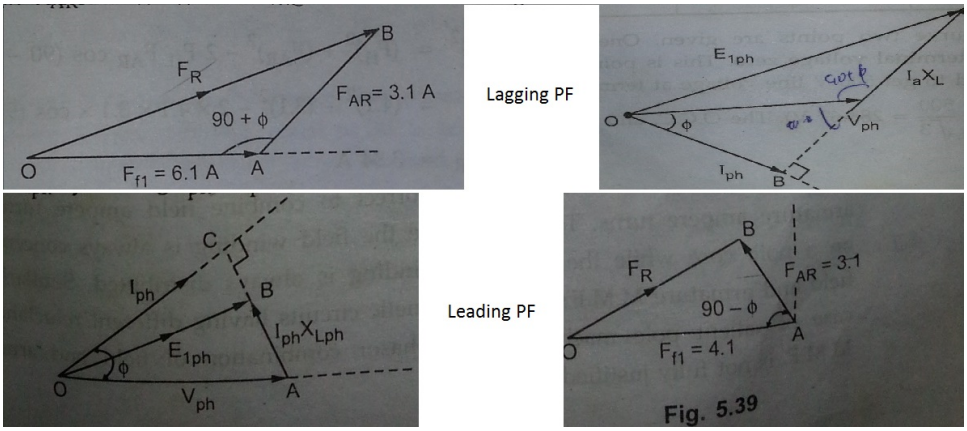


Figure 5.5: TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTORS

```
4  P=1200*10^3
5  V_line=12000
6  R_a=2,X_s=35//armature resistance and synchronous
       reactance
7  phi=acos(0.8)
8
9
10 I_L=P/(sqrt(3)*V_line*cos(phi))
11 I_a=I_L
12 V_ph=V_line/sqrt(3)
13 E_ph=sqrt((V_ph*cos(phi)+I_a*R_a)^2+(V_ph*sin(phi)+
       I_a*X_s)^2)
14 regulation=100*(E_ph-V_ph)/V_ph
15
16 printf('Regulation at 0.8 lag power factor is %.2f
       percent',regulation)
```

Scilab code Exa 5.9 TO DETERMINE FULL LOAD REGULATION AT VARIOUS POWER FACTORS

```
clc,clear
printf('Example 5.9\n\n')

V_L=11000  , V_ph= V_L/sqrt(3)
VA=1000*1000
I_L=VA/(V_L*sqrt(3))

V_OC_ph=433/sqrt(3)
I_asc_ph=I_L

Z_s=V_OC_ph /I_asc_ph  //ohms per phase
R_a=0.45 //ohms per phase
X_s=sqrt(Z_s^2-R_a^2)

//part(i)
phi=acos(0.8) //lagging
E_ph = sqrt((V_ph*cos(phi)+I_L*R_a)^2 +(V_ph*sin(phi
   )+ I_L*X_s)^2)
regulation=100*(E_ph-V_ph)/V_ph
printf('Voltage regulation at 0.8 pf lagging is %f
   percent\n',regulation)

//part(ii)
phi=acos(0.8) //leading
E_ph2 = sqrt((V_ph*cos(phi)+I_L*R_a)^2 +(V_ph*sin(
   phi)- I_L*X_s)^2)
regulation2=100*(E_ph2-V_ph)/V_ph
printf('Voltage regulation at 0.8 pf lagging is %f
   percent\n',regulation2)
printf('\nAnswer mismatches due to improper
   approximation')
```

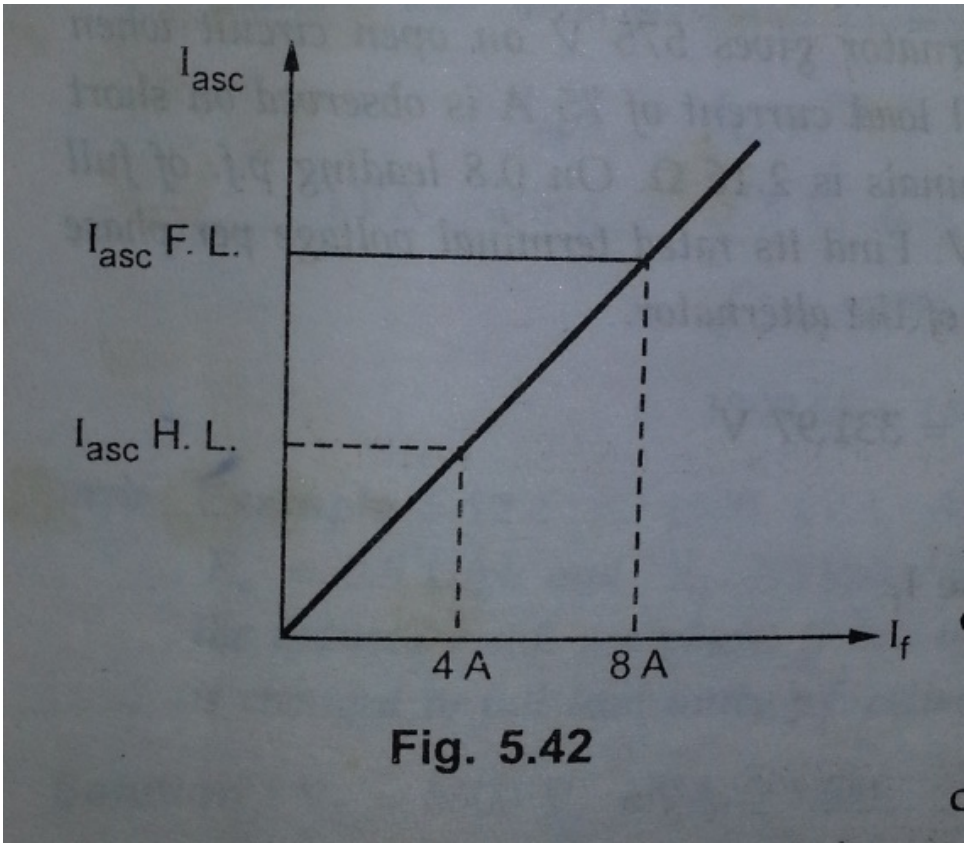


Figure 5.6: TO CALCULATE PERCENTAGE REGULATION FOR HALF LOAD

Scilab code Exa 5.10 TO CALCULATE PERCENTAGE REGULATION FOR HALF LOAD

```
clc,clear
printf('Example 5.10\n\n')

VA=125*10^3
V_L=400,V_ph=V_L/sqrt(3)
I_L=VA/(sqrt(3)*V_L)
I_aph=I_L

I_f=4,I_asc=I_aph/2 //for half load.. refer to graph
V_OC_line=140
V_OC_ph=V_OC_line/sqrt(3)
I_asc_ph=I_asc
Z_s= V_OC_ph/I_asc_ph
R_a=0.1,X_s=sqrt(Z_s^2-R_a^2) //armature resistance
   and synchronous reactance

phi=acos(0.8)
E_ph = sqrt((V_ph*cos(phi)+I_asc*R_a)^2 +(V_ph*sin(
   phi)- I_asc*X_s)^2)
regulation=100*(E_ph-V_ph)/V_ph
printf('Voltage regulation at 0.8 pf leading for
   half load is %.2f percent\n',regulation)
```

Scilab code Exa 5.11 TO DETERMINE RATED TERMINAL VOLTAGE AND kVA RATING OF ALTERNATOR

```
clc,clear
printf('Example 5.11\n\n')

V_OC_line=575,   V_OC_ph=V_OC_line/sqrt(3)
I_asc_line=75
I_asc_ph  =I_asc_line
```

```
I_aph=I_asc_ph
I_L=I_aph

Z_s= V_OC_ph/I_asc_ph
R_a=2.16/2
X_s =  sqrt(Z_s^2 - R_a^2)

//on full load
E_ph=6100
phi=acos(0.8) //leading

//using E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(V_ph
   *sin(phi)- I_a*X_s)^2)
p=[1 -256.68 -3.71*10^7]
roots(p)
V_ph=ans(1) //second root is ignored as its -ve
V_L=V_ph*sqrt(3)
printf('Rated terminal voltage between the lines is
   %.3f V \n',V_L)
VA_rating=sqrt(3)*V_L*I_L
printf('kVA rating of the alternator is %.2f kVA',
   VA_rating*10^-3)
```

Scilab code Exa 5.12 TO DETERMINE INDUCED EMF AND TERMINAL VOLTAGE PER PHASE

```
clc,clear
printf('Example 5.12\n\n')

V_L=6600, V_ph=V_L/sqrt(3)
VA=1500*10^3
I_L=VA/(sqrt(3)*V_L)
I_aph=I_L

R_a=0.5, X_s=5//armature resistance and synchronous
```

```
    reactance
phi=acos(0.8)
E_ph = sqrt((V_ph*cos(phi)+I_aph*R_a)^2 +(V_ph*sin(
    phi)+ I_aph*X_s)^2)
printf('Induced EMF per phase is %f V\n',E_ph)

//full load
phi=acos(1)
//using E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(V_ph
    *sin(phi)- I_a*X_s)^2)
p=[1 131.215 -1.791*10^7]
roots(p)
V_ph=ans(2) //first root is ignored as it is -ve
printf('Terminal voltage per phase is %f V',V_ph)
```

Scilab code Exa 5.13 TO DETERMINE VOLTAGE REGULATION BY EMF METHOD AT VARIOUS POWER FACTORS

```
clc,clear
printf('Example 5.13\n\n')

V_ph=2000
R_a=0.8
I_sc=100
V_OC=500
I_f=2.5
Z_s=V_OC/I_sc
X_s= sqrt(Z_s^2 - R_a^2 )
I_aFL=100,I_a=I_aFL

//part(i)
phi=acos(1)
E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(V_ph*sin(phi
    )+ I_a*X_s)^2)
regulation=100*(E_ph-V_ph)/V_ph
```

```
17 printf('(i)Voltage regulation is %.2f percent\n',
     regulation)
18
19 //part(ii)
20 phi2=acos(0.8)
21 E_ph2 = sqrt((V_ph*cos(phi2)+I_a*R_a)^2 +(V_ph*sin(
     phi2)- I_a*X_s)^2)
22 regulation2=100*(E_ph2-V_ph)/V_ph
23 printf('(ii)Voltage regulation is %.2f percent\n',
     regulation2)
24
25 //part(iii)
26 phi3=acos(0.71)
27 E_ph3 = sqrt((V_ph*cos(phi3)+I_a*R_a)^2 +(V_ph*sin(
     phi3)+ I_a*X_s)^2)
28 regulation3=100*(E_ph3-V_ph)/V_ph
29 printf('(iii)Voltage regulation is %.2f percent\n',
     regulation3)
```

Scilab code Exa 5.14 TO FIND FULLLOAD VOLTAGE REGULATION USING SYNCHRONOUS IMPEDANCE METHOD

```
1 clc,clear
2 printf('Example 5.14\n\n')
3
4 VA=1000*1000
5 V_L=4600 , V_ph=V_L/sqrt(3)
6 I_L=VA/(sqrt(3)*V_L)
7 I_aph_FL=I_L,I_aph=I_aph_FL
8 I_sc=(150/100)* I_aph_FL
9 V_OC_line=1744
10 V_OC_ph= V_OC_line/sqrt(3)
11
12 Z_s=V_OC_ph / I_sc
13 R_a=1
```

```
14 X_s=sqrt(Z_s^2-R_a^2)
15
16 phi=acos(0.8 ) //lagging
17 E_ph = sqrt((V_ph*cos(phi)+I_aph*R_a)^2 +(V_ph*sin(
     phi)+ I_aph*X_s)^2)
18 regulation=100*(E_ph-V_ph)/V_ph
19 printf('Voltage regulation at full load 0.8 pf is %
     .2f percent\n',regulation)
```

Scilab code Exa 5.15 TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD

```
1 clc,clear
2 printf('Example 5.15\n\n')
3
4 //part(i)    Ampere turn method
5 F_O=37.5
6 F_AR=20
7 V_L=6600, V_ph=V_L/sqrt(3)
8
9 //lagging
10 phi=acos(0.8)
11 F_R= sqrt((F_O+F_AR*sin(phi) )^2 + (F_AR*cos(phi))^2
     )
12 //E_ph corresponding to F_R can be obtained by
     plotting open circuit characteristics
13 E_ph=4350
14 regulation=100*(E_ph-V_ph)/V_ph
15 printf('(i)By Ampere-turn method or MMF method\nFull
     -load regulation at 0.8 lagging pf is %.2f
     percent\n',regulation)
16 //leading
17 phi=acos(0.8)
18 F_R= sqrt((F_O-F_AR*sin(phi) )^2 + (F_AR*cos(phi))^2
     )
```

```
19 //E_ph corresponding to F_R can be obtained by
      plotting open circuit characteristics
20 E_ph=3000
21 regulation=100*(E_ph-V_ph)/V_ph
22 printf('Full-load regulation at 0.8 leading pf is %
      .2f percent\n',regulation)
23
24 //EMF method
25 V_OC_ph=100,V_ph=100
26 I_sc= 100*(F_O/F_AR)    //times the rated value
27 Z_s=V_OC_ph/I_sc
28 F_O= 100
29 F_AR= Z_s*100
30
31 //lagging
32 phi=acos(0.8)
33 F_R=  sqrt((F_O+F_AR*sin(phi) )^2 + (F_AR*cos(phi))
      ^2 )
34 regulation=100*(F_R-V_ph)/V_ph
35 printf('\n(ii)Synchronous impedance method or EMF
      method\n')
36 printf('Full-load regulation at 0.8 lagging pf is %
      .2f percent\n',regulation)
37 //leading
38 phi=acos(0.8)
39 F_R=  sqrt((F_O-F_AR*sin(phi) )^2 + (F_AR*cos(phi))
      ^2 )
40 regulation=100*(F_R-V_ph)/V_ph
41 printf('Full-load regulation at 0.8 leading pf is %
      .2f percent\n',regulation)
```

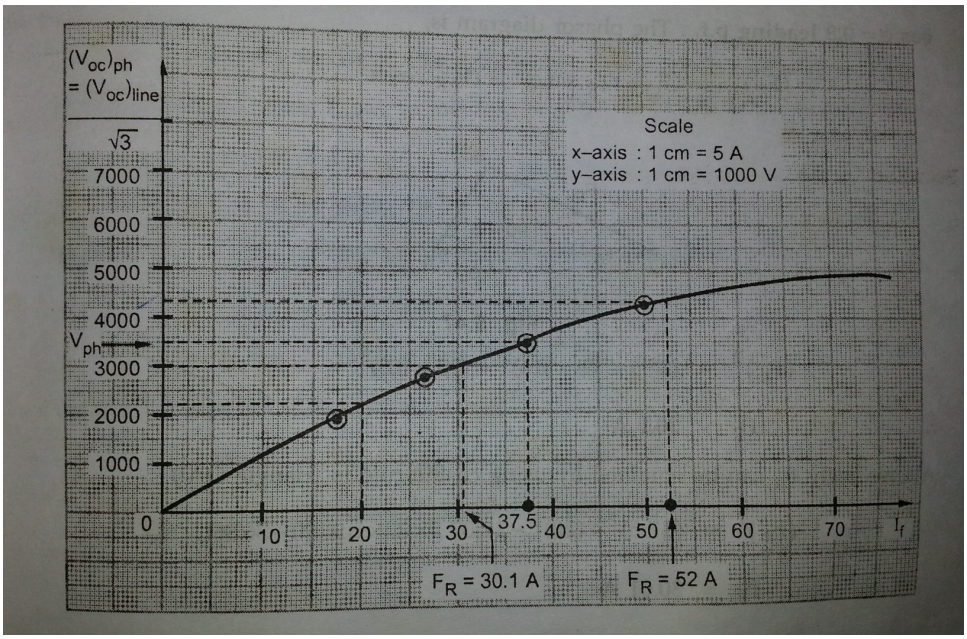


Figure 5.7: TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD

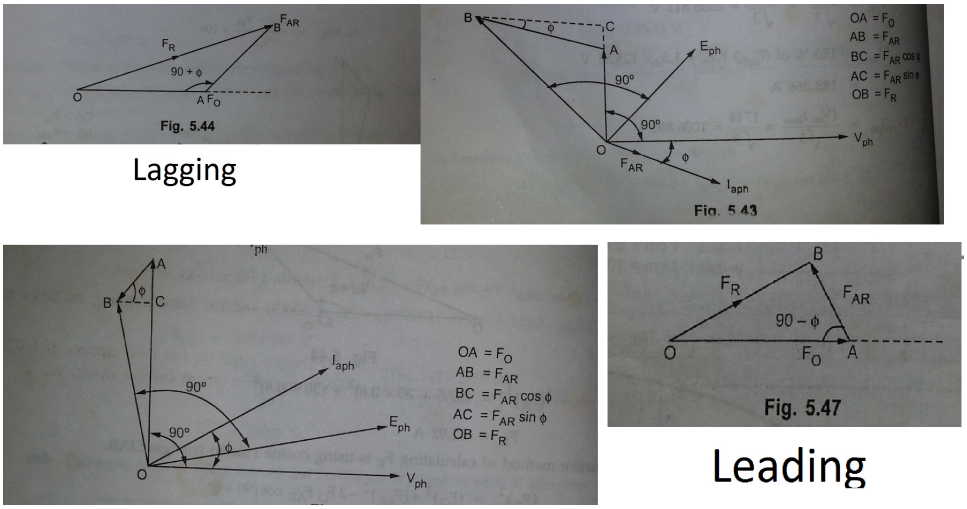


Figure 5.8: TO CALCULATE FULL LOAD REGULATION BY MMF AND SYNCHRONOUS IMPEDANCE METHOD

Scilab code Exa 5.16 TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD

```
clc,clear
printf('Example 5.16\n\n')

V_L=6000,V_ph=V_L/sqrt(3)
I_ph_X_Lph = 0.9*500  //leakage reactance drop in
   volts = 0.9 cm * 500 V/cm
phi= acos(0.8) //lagging

E_1ph=sqrt( (V_ph*cos(phi))^2 + (V_ph*sin(phi)+
   I_ph_X_Lph)^2 )   //From triangle OAB
F_f1 = 26 //from OCC
F_AR= 2.9*5  //2.9cm * 5 A/cm

F_R =   sqrt(F_f1^2 + F_AR^2 -2*F_AR*F_f1*cos(phi+ (
   %pi/2)) )
printf('Required field current is %.2f A',F_R)
```

Scilab code Exa 5.17 TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE

```
clc,clear
printf('Example 5.17\n\n')

V_L=400,V_ph=V_L/sqrt(3)
VA=40*10^3
I_L=VA/(sqrt(3)*V_L) , I_aph=I_L
I_aph_X_Lph = 0.65*50  //leakage reactance drop in
   volts = 2.4 cm * 500 V/cm
```

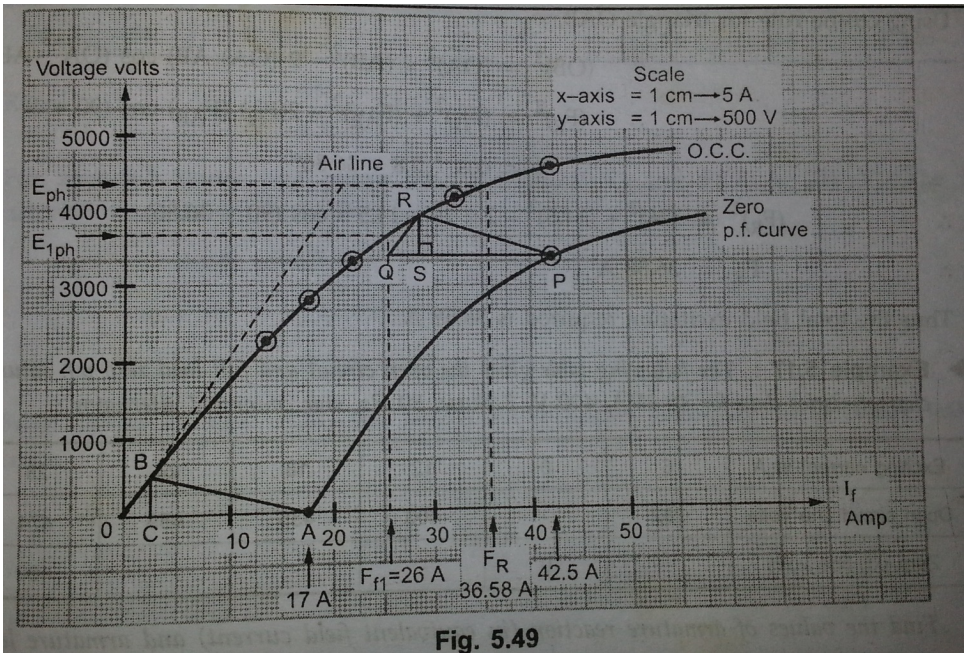
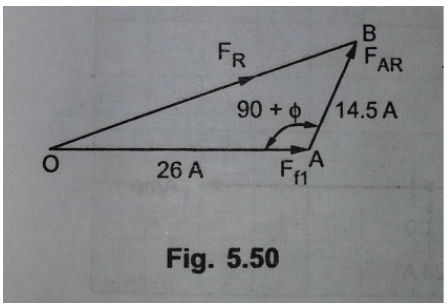


Figure 5.9: TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD



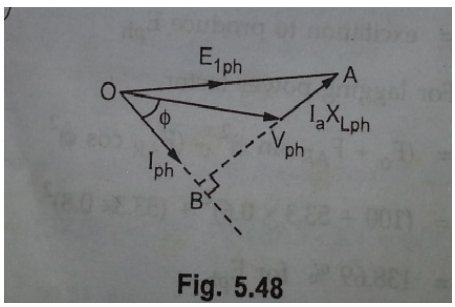


Figure 5.10: TO DETERMINE FIELD CURRENT REQUIRED DURING FULL LOAD

```
X_Lph= I_aph_X_Lph/ I_aph
printf('Armature leakage reactance is %.3f ohms\
   nNote:This answer doesnt match with textbook as
   it has been reciprocated in textbook\n\n',X_Lph)
phi=acos(0.8)  //lagging
E_ph = sqrt((V_ph*cos(phi))^2 +(V_ph*sin(phi)+
   I_aph_X_Lph)^2)
F_f1=15.6 //as obtained from OCC corresponding to
   this E_ph

F_AR= 2.3*3  //2.3cm * 3 A/cm
printf('Armature reaction is %.1f \n',F_AR)
F_R =   sqrt(F_f1^2 + F_AR^2 -2*F_AR*F_f1*cos(phi+ (
   %pi/2)) )  //cosine rule to Triangle OAB
E_ph=267.5  //corresponding to F_R from open circiut
    characteristics
regulation=100*(E_ph-V_ph)/V_ph
printf('Voltage regulation at 0.8 pf lagging is %.1f
    percent\n',regulation)
```

Scilab code Exa 5.18 TO FIND VOLTAGE REGULATION OF ALTERNATOR FOR FULL LOAD CURRENT USING POTIER METHOD

```
clc,clear
printf('Example 5.18\n\n')

VA=10*10^3
V_L=11*10^3,V_ph=V_L/sqrt(3)
I_ph_X_Lph = 2.4*500  //leakage reactance drop in
```

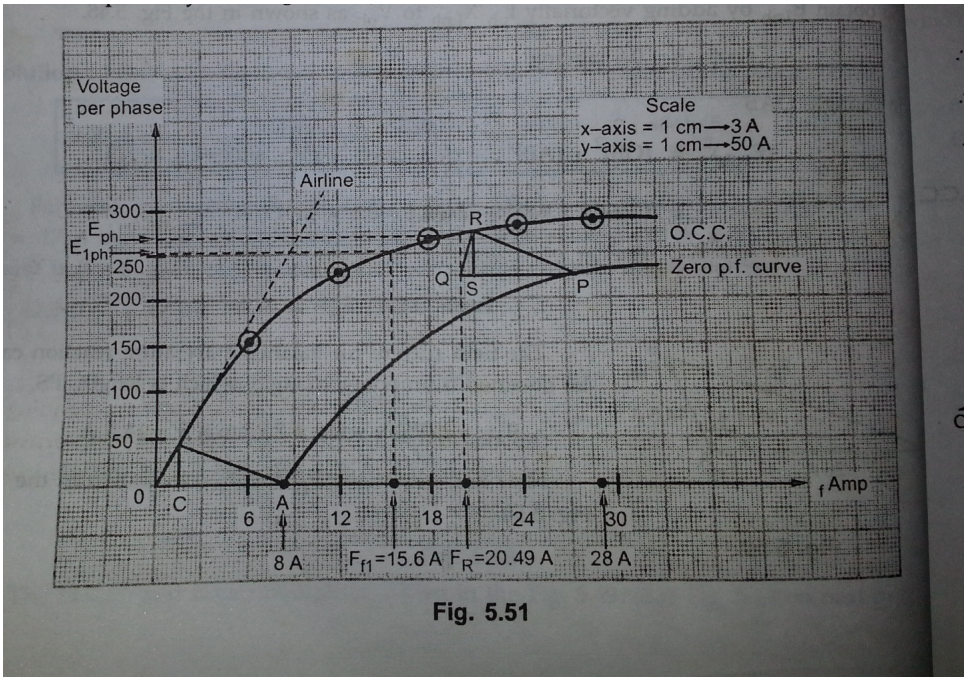


Fig. 5.51

Figure 5.11: TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE

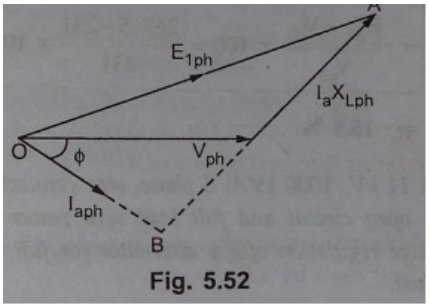


Fig. 5.52

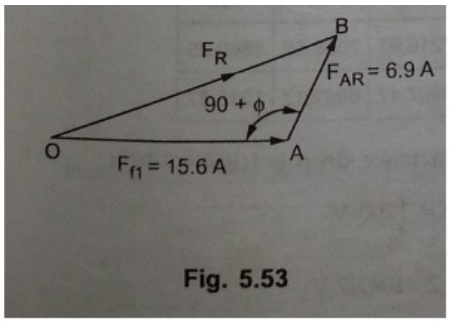


Fig. 5.53

Figure 5.12: TO DETERMINE VOLTAGE REGULATION ARMATURE REACTION AND LEAKAGE RESISTANCE

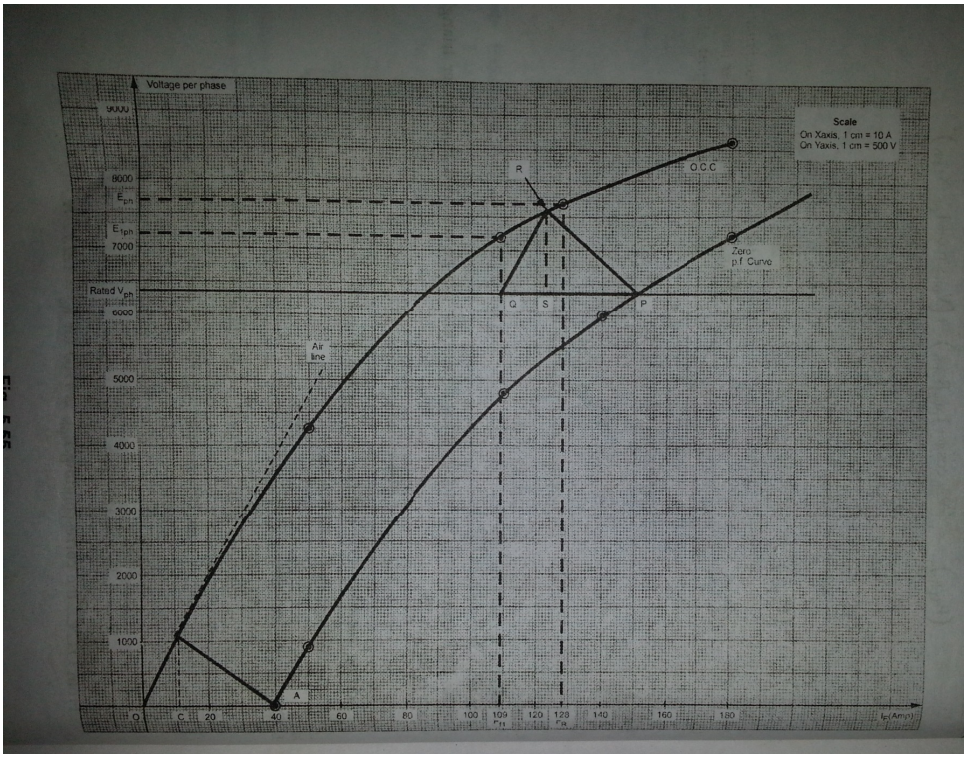


Figure 5.13: TO FIND VOLTAGE REGULATION OF ALTERNATOR FOR FULL LOAD CURRENT USING POTIER METHOD

```
    volts = 2.4 cm * 500 V/cm
I_ph_R_aph =VA/(sqrt(3)*V_L)
phi=acos(0.8)
E_ph = sqrt((V_ph*cos(phi)+I_ph_R_aph)^2 +(V_ph*sin(
    phi)+ I_ph_X_Lph)^2)
F_f1=109 //obtained from open circuit
    characteristics corresponding to calculated E_ph
F_AR= 2.8*10  //2.8cm * 10 A/cm
F_R =   sqrt(F_f1^2 + F_AR^2 -2*F_AR*F_f1*cos(phi+ (
    %pi/2)) )  //cosine rule to Triangle OAB
E_ph=7700  //corresponding to F_R from open circiut
    characteristics

regulation=100*(E_ph-V_ph)/V_ph
printf('Voltage regulation at full-load 0.8 pf
    lagging is %.2f percent\n',regulation)
```

Scilab code Exa 5.19 TO DETERMINE TERMINAL VOLTAGE AT A GIVEN EXCITATION

```
clc,clear
printf('Example 5.19\n\n')

VA=2000*1000
V_L=11000,V_ph=V_L/sqrt(3)
R_a=0.3,X_s=5 //armature resistance and synchronous
    reactance

//case (i)
phi=acos(0.8) //lagging
I_L=VA/(sqrt(3)*V_L)  , I_a=I_L
E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(V_ph*sin(phi
    )+ I_a*X_s)^2)

//Using E_ph = sqrt((V_ph*cos(phi)+I_aph*R_a)^2 +(
```

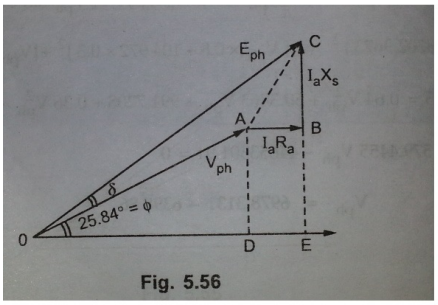
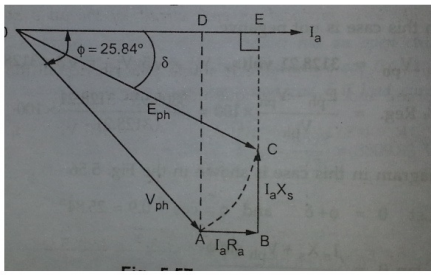


Fig. 5.56



0.9 lagging

0.8 leading

Figure 5.14: TO DETERMINE TERMINAL VOLTAGE LOAD ANGLE AND VOLTAGE REGULATION

```
    V_ph*sin(phi)+ I_aph*X_s)^2)
14 //we get  V_ph^2 -579.4455 V_ph -44653301.91=0
15 p=[1 -579.4455 -44653301.91]
16 roots(p)
17 V_ph=ans(1) //second root is ignored as its -ve
18 printf('Terminal voltage is %.4f V',V_ph)
```

Scilab code Exa 5.20 TO DETERMINE TERMINAL VOLTAGE LOAD ANGLE AND VOLTAGE REGULATION

```
1 clc,clear
2 printf('Example 5.20\n\n')
3
4 R_a=0.6,X_s=6 //armature resistance and synchronous
    reactance per phase
5 E_L=6599, E_ph=E_L/sqrt(3)
6 I_L=180,I_a=I_L
7
```

```
8  //part(i)
9  // using E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(
       V_ph*sin(phi)+ I_a*X_s)^2) and solving for V_ph
10 p=[1 1135.83 -13338836.49]
11 roots(p)
12 V_ph=ans(2)
13 V_L=V_ph*sqrt(3)
14 regulation=100*(E_ph-V_ph)/V_ph
15
16 phi=acos(0.9)
17 theta=atan(       (I_a*X_s+V_ph*sin(phi) )/(E_ph)
                   )
18 delta=theta-phi
19 printf('(i)0.9 lagging\nTerminal voltage is %.2f V\
       nVoltage regulation is %.2f percent\nLoad angle
       is %.2f degrees',V_ph*sqrt(3),regulation,delta
       *(180/%pi))
20
21 //part(ii)
22 phi_2=acos(0.8)
23 // using E_ph = sqrt((V_ph*cos(phi)+I_a*R_a)^2 +(
       V_ph*sin(phi)- I_a*X_s)^2) and solving for V_ph
24 p=[1 -941.53 -11399574.87]
25 roots(p)
26 V_ph=ans(1) //second root is ignored as its -ve
27 V_L=V_ph*sqrt(3)
28 regulation2=100*(E_ph-V_ph)/V_ph
29 delta_2 = asin( (tan(phi)*(V_ph*cos(phi_2)+I_a*R_a)
       -I_a*X_s )/E_ph    )
30 printf('\n\n(ii)0.8 leading\nTerminal voltage is %.2
       f V\nVoltage regulation is %.2f percent\nLoad
       angle is %.2f degrees',V_L,regulation2,delta_2
       *(180/%pi))
```

Scilab code Exa 5.21 TO DETERMINE VOLTAGE REGULATION BY EMF METHOD AT VARIOUS POWER FATORS

```
clc,clear
printf('Example 5.21\n\n')

V_ph=2000
R_a=0.8
I_sc=100,I_a=I_sc
V_OC=500
I_f=2.5
Z_s=V_OC/I_sc

X_s=sqrt(Z_s^2- R_a^2)
I_a_FL=100

//Part(i)
phi1=acos(1) //and lagging
E_ph1=sqrt((V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(phi1
   )+I_a*X_s)^2)
regulation1=100*(E_ph1-V_ph)/V_ph
printf('Regulation at upf is %.2f percent\n',
   regulation1)

//Part(ii)
phi2=acos(0.8)
E_ph2=sqrt((V_ph*cos(phi2)+I_a*R_a)^2+(V_ph*sin(phi2
   )-I_a*X_s)^2)
regulation2=100*(E_ph2-V_ph)/V_ph
printf('Regulation at 0.8 leading pf is %.2f percent
   \n',regulation2)

//Part(iii)
phi3=acos(0.71)
E_ph3=sqrt((V_ph*cos(phi3)+I_a*R_a)^2+(V_ph*sin(phi3
   )+I_a*X_s)^2)
regulation3=100*(E_ph3-V_ph)/V_ph
printf('Regulation at 0.71 lagging pf is %.2f
```

```
percent\n',regulation3)
```

Scilab code Exa 5.22 TO DETERMINE CERTAIN QUANTITIES ASSOCIATED WITH SINGLE PHASE ALTERNATOR

```
clc,clear
printf('Example 5.22\n\n')

V=600
VA=60*10^3
I_sc=210
V_oc=480
I_f=10
R_a=0.2

I=VA/V          //VA=V*I and alternator is single
   phase
I_a=I

Z_s=V_oc/I_sc  //Synchronous Impedance
X_s=sqrt(Z_s^2-R_a^2) //SYnchronous Reactance
printf('Synchronous impedances is %f ohms and
   synchronous reactance is %f ohms\n',Z_s,X_s)

//PART (i)
phi1=acos(0.8) //and lagging
E1=sqrt((V*cos(phi1)+I_a*R_a)^2+(V*sin(phi1)+I_a*X_s
   )^2) //plus sign for lagging power factor
regulation1=100*(E1-V)/V
printf('\nRegulation at 0.8 lagging pf is %.2f
   percent ',regulation1 )

//PART (ii)
phi2=acos(1)
E2=sqrt((V*cos(phi2)+I_a*R_a)^2+(V*sin(phi2)+I_a*X_s
```

```
    )^2)
27 regulation2=100*(E2-V)/V
28 printf('\nRegulation at UNITY pf is %.2f percent ',
    regulation2 )
29
30 //PART (iii)
31 phi3=acos(0.6) //and leading
32 E3=sqrt((V*cos(phi3)+I_a*R_a)^2+(V*sin(phi3)-I_a*X_s
    )^2) //minus sign for leading power factor
33 regulation3=100*(E3-V)/V
34 printf('\nRegulation at 0.6 leading pf is %.2f
    percent ',regulation3 )
```

Scilab code Exa 5.23 TO DETERMINE FULL LOAD VOLTAGE REGULATION AT LEADING AND LAGGING POWER FACTOR

```
1  clc,clear
2  printf('Example 5.23\n\n')
3
4  V_L=3300, V_ph=V_L/sqrt(3)
5  I_a=100
6  I_f=5
7  V_OC_line=900 ,V_OC_ph=V_OC_line/sqrt(3)
8  R_a=0.8 //armature resistance
9  I_aph=I_a
10 Z_s=V_OC_ph/I_aph
11 X_s=sqrt(Z_s^2-R_a^2) //synchronous reactance
12
13 //Part(i)
14 phi1=acos(0.8) //and lagging
15 E_ph1=sqrt((V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(phi1
    )+I_a*X_s)^2)
16 regulation1=100*(E_ph1-V_ph)/V_ph
17 printf('Regulation at 0.8 lagging is %.2f percent\n'
    ,regulation1)
```

```
18
19 //Part(ii)
20 phi2=acos(0.8)  //and leading
21 E_ph2=sqrt((V_ph*cos(phi2)+I_a*R_a)^2+(V_ph*sin(phi2
   )-I_a*X_s)^2)
22 regulation2=100*(E_ph2-V_ph)/V_ph
23 printf('Regulation at 0.8 leading pf is %.2f percent
   \n',regulation2)
```

Scilab code Exa 5.24 TO CALCULATE PERCENTAGE REGULATION AT LEADING LAGGING AND UNITY POWER FACTORS

```
1 clc,clear
2 printf('Example 5.24\n\n')
3
4 V_L=13500
5 R_a=1.5 , X_s=30 //armature resistance and
   synchronous reactance
6 V_ph=V_L/sqrt(3)
7
8 //CASE 1
9 phi1=acos(0.8)
10 P_out=1280*10^3
11 I_L= P_out/ (sqrt(3)*V_L*cos(phi1) )    //because
   P_out=sqrt(3)*V_L*I_L*cos(phi)
12
13 I_a=I_L
14 E_ph=sqrt((V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(phi1)
   +I_a*X_s)^2)
15 regulation=100*(E_ph-V_ph)/V_ph
16 printf('Regulation at 0.8 lagging power factor is %
   .2f percent',regulation)
17
18 //Case 2
19 phi2=acos(1)
```

```
20 I_L= P_out/ (sqrt(3)*V_L*cos(phi2) )     //because
     P_out=sqrt(3)*V_L*I_L*cos(phi)
21
22 I_a=I_L
23 E_ph=sqrt((V_ph*cos(phi2)+I_a*R_a)^2+(V_ph*sin(phi2)
     +I_a*X_s)^2)
24 regulation2=100*(E_ph-V_ph)/V_ph
25 printf('\nRegulation at unity power factor is %.2f
     percent',regulation2)
26
27 //case 3
28 phi3=acos(0.8)
29 I_L= P_out/ (sqrt(3)*V_L*cos(phi3) )     //because
     P_out=sqrt(3)*V_L*I_L*cos(phi)
30 I_a=I_L
31 E_ph=sqrt((V_ph*cos(phi3)+I_a*R_a)^2+(V_ph*sin(phi3)
     -I_a*X_s)^2)// minus sign in the second bracket
     beacuse of leading pf
32 regulation3=100*(E_ph-V_ph)/V_ph
33 printf('\nRegulation at 0.8 leading power factor is
     %.2f percent',regulation3)
```

Scilab code Exa 5.26 TO CALCULATE PERCENTAGE REGULATION USING EMF METHOD

```
1 clc,clear
2 printf('Example 5.26\n\n')
3
4 V_L=11*10^3
5 VA_rating=10^6
6 R_a=2.2 //alternator resistance
7 phi=acos(0.8)
8
9 I_L=VA_rating/(sqrt(3)*V_L)//VA=sqrt(3)V_L*I_L
10 I_a=I_L
```

```
11 V_ph=V_L/sqrt(3)
12 regulation=24
13
14 E_ph= ((regulation/100)+1)*V_ph     // because
      regulation=100*(E_ph-V_ph)/V_ph
15 //using E_ph=sqrt((V_ph*cos(phi)+I_a*R_a)^2+(V_ph*
      sin(phi)+I_a*X_s)^2)
16 X_s=(sqrt(E_ph^2-((V_ph*cos(phi)+I_a*R_a)^2))-V_ph*
      sin(phi))*(1/I_a)
17
18 phi1=acos(0.8)
19 E_ph=sqrt((V_ph*cos(phi1)+I_a*R_a)^2+(V_ph*sin(phi1)
      -I_a*X_s)^2)
20 regulation1=100*(E_ph-V_ph)/V_ph
21 printf('\nRegulation at 0.8 leading power factor is
      %.2f percent',regulation1)
```

Scilab code Exa 5.27 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO STAR CONNECTED ALTERNATOR

```
1  clc,clear
2  printf('Example 5.27\n\n')
3
4  V_L=220
5  VA=100*10^3
6  R_a=0.1 //effective resistacne of alternator
7  X_a=0.5 //leakage reactance
8  X_ar=2*X_a
9
10 Z_s=complex(R_a,X_a+X_ar)
11
12 //Part(1)
13 phi=acos(0.4)
14 V_ph=V_L/sqrt(3)
15 I_L=VA/(sqrt(3)*V_L)//VA=sqrt(3)*V_L*I_L
```

```
16 I_a=I_L
17 E_ph=sqrt((V_ph*cos(phi)+I_a*R_a)^2+(V_ph*sin(phi)+
      I_a*(X_a+X_ar))^2)
18 printf('(i)Required noload voltage is %.3f V',E_ph)
19
20 //Part(2)
21 V_ph2=0
22 E_ph2=sqrt((V_ph2*cos(phi)+I_a*R_a)^2+(V_ph2*sin(phi
      )+I_a*(X_a+X_ar))^2)
23 printf('\n(ii)Required noload voltage is %.3f V',
      E_ph2)
```

Scilab code Exa 5.28 TO DETERMINE FULL LOAD PERCENTAGE REGULATION AT A LEADING AND LAGGING POWER FACTOR

```
1 clc,clear
2 printf('Example 5.28\n\n')
3
4 V_L=2000,V_ph=V_L/sqrt(3)
5 VA=1000*10^3
6 I_L=VA/(sqrt(3)*V_L)     //because VA=sqrt(3)*V_L*I_L
7 I_aph=I_L
8
9 I_f=28.5//for this I_aph=288.67513 as obtained from
      SCC graph
10 V_oc_ph=1060//for I_f=28.5 as obtained fromOCC graph
11 Z_s=V_oc_ph/I_aph
12 R_a=0.2 //armature effective resistance
13 X_s=sqrt( Z_s^2-R_a^2 )
14
15 //Part(i)
16 phi1=acos(0.8)//lagging
17 E_ph1=sqrt((V_ph*cos(phi1)+I_aph*R_a)^2+(V_ph*sin(
```

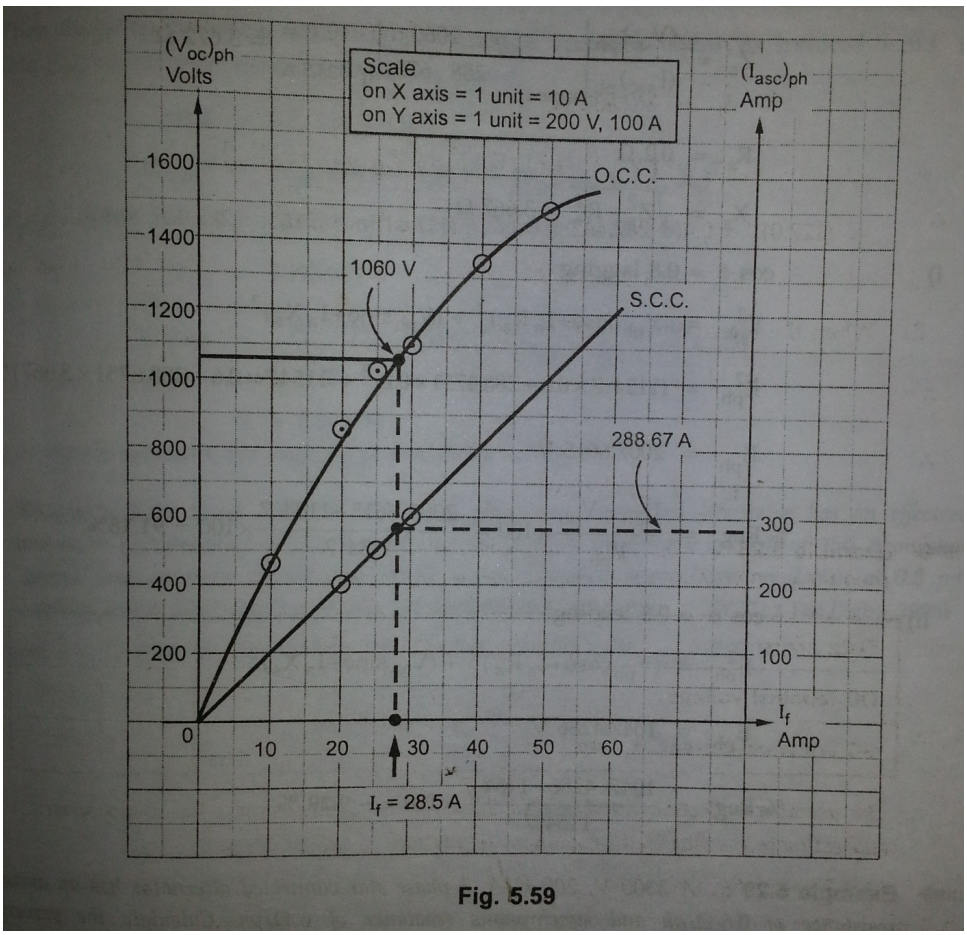


Figure 5.15: TO DETERMINE FULL LOAD PERCENTAGE REGULATION AT A LEADING AND LAGGING POWER FACTOR

```
   phi1)+I_aph*X_s)^2)
18 regulation1=100*(E_ph1-V_ph)/V_ph
19 printf("(i)Full-load percentage regulation at 0.8 pf
    lagging is %.2f percent",regulation1)
20
21 //Part(ii)
22 phi2=acos(0.8)//leading
23 E_ph2=sqrt((V_ph*cos(phi2)+I_aph*R_a)^2+(V_ph*sin(
   phi2)-I_aph*X_s)^2)
24 regulation2=100*(E_ph2-V_ph)/V_ph
25 printf("\n(ii)Full-load percentage regulation at 0.8
    pf leading is %.2f percent\n\n",regulation2)
26 printf('Note that the answer mismatches because of
   calculation mistake done in the last step of part
    1')
```

Scilab code Exa 5.29 TO CALCULATE PERCENTAGE REGULATION WHEN RATED OUTPUT SWITCHES OFF

```
1 clc,clear
2 printf('Example 5.29\n\n')
3
4 V_L=3300
5 VA=200*10^3
6 R_a=0.6, X_s=6//armature resistance and synchronous
   reactance
7
8 I_L=VA/(sqrt(3)*V_L)//VA=sqrt(3)V_L*I_L
9 I_a=I_L
10 V_ph=V_L/sqrt(3)
11 phi=acos(0.8)
12 E_ph=sqrt((V_ph*cos(phi)+I_a*R_a)^2+(V_ph*sin(phi)+
   I_a*X_s)^2)
13
14 regulation=100*(E_ph-V_ph)/V_ph
```

```
15 printf('Regulation at 0.8 lagging power factor is %
      .3f percent',regulation)
16 printf('\n Note :\n Regulation is positive for
      lagging power factor loads')
```

Scilab code Exa 5.30 TO CALCULATE VOLTAGE REGULATION FOR FULL LOAD CURRENT AT CERTAIN LEADING AND LAGGING POWER FACTORS

```
1  clc,clear
2  printf('Example 5.30\n\n')
3
4  V_L=2300    ,  V_ph= V_L/sqrt(3)
5  f=50,R_a=0.2 //armature resistance
6  I_sc=150
7  V_OC_line=780    , V_OC_ph=V_OC_line/sqrt(3)
8
9  Z_s= V_OC_ph/I_sc
10 X_s =  sqrt(Z_s^2 - R_a^2)
11 I_aph=25 , I_aFL=I_aph
12
13 //part(i)
14 phi= acos(0.8)  //lag
15 E_ph = sqrt((V_ph*cos(phi)+I_aph*R_a)^2 +(V_ph*sin(
      phi)+ I_aph*X_s)^2)
16 regulation=100*(E_ph-V_ph)/V_ph
17 printf('Voltage regulation at 0.8 pf lagging is %.3f
       percent\n',regulation)
18
19 //part(ii)
20 phi2= acos(0.8)  //lead
21 E_ph2 = sqrt((V_ph*cos(phi2)+I_aph*R_a)^2 +(V_ph*sin
      (phi2)- I_aph*X_s)^2 )
22 regulation2=100*(E_ph2-V_ph)/V_ph
23 printf('Voltage regulation at 0.8 pf leading is %.3f
```

```
percent',regulation2)
```

Chapter 6

Synchronization and Parallel Operation of Alternators

Scilab code Exa 6.2 TO DETERMINE TOTAL INDUCED EMF ON OPEN CIRCUIT

```
1 clc,clear
2 printf('Example 6.2\n\n')
3
4 X_d=0.7 , X_q=0.4  //direct and quadrature axis
    synchronous reactance p.u.
5 R_a=0
6 phi=acos(0.8)  //Lag
7
8 V_t=1 //assumed rated terminal Voltage
9 I_a=1 //Full-load armature current
10
11 psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
    R_a)            )
12 delta=psi-phi
13 I_d=I_a*sin(psi)
14 I_q=I_a*cos(psi)
15 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
16 printf('Total e.m.f induced on open circuit is %.4f
```

```
p.u.',E_f)
```

Scilab code Exa 6.3 TO DETERMINE CERTAIN CHARACTERISTICS OF SINGLE PHASE ALTERNATORS WORKING IN PARALLEL

```
clc,clear
printf('Example 6.3\n\n')

//note that a new function p2z has been defined
   below for direct representation of complex
   numbers in polar form
function [FUN] = p2z(RRRR,Theeeta)
 FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
endfunction

Z1=complex(0,3) //impedance of alternator 1
Z2=complex(0,4) //impedance of alternator 2
Z=6 //load
E1=p2z(220,0) //induced emf vector on no load
E2=p2z(220,10)//induced emf vector on no load

I1=((E1-E2)*Z+E1*Z2)/(Z*(Z1+Z2)+Z1*Z2)
I2=((E2-E1)*Z+E2*Z1)/(Z*(Z1+Z2)+Z1*Z2)

phi1=phasemag(I1)  //Phasemag returns the angle of
   complex number in degrees
phi2=phasemag(I2)  //Phasemag returns the angle of
   complex number in degrees

I=I1+I2
V=I*Z //Terminal voltage
printf('(i) Terminal voltage is %.1f volts at %.2f
   degrees\n',abs(V),phasemag(V))
printf('(ii) Currents are %.2f A at %.2f degrees and
    %.2f A at %.2f degrees\n     Total current is %
```

```
    .2f A at %.2f degrees  ',abs(I1),phasemag(I1),abs
    (I2),phasemag(I2),abs(I),phasemag(I))
25
26 P1=abs(V)*abs(I1)*cosd(phi1)
27 P2=abs(V)*abs(I2)*cosd(phi2)
28 printf('\n(iii)Power delivered is %.2f watts and %.2
    f watts',P1,P2)
```

Scilab code Exa 6.4 TO CALCULATE SYNCHRONISING POWER OF ARMATURE PER MECHANICAL DEGREE OF PHASE DISPLACEMENT

```
1  clc,clear
2  printf('Example 6.4\n\n')
3
4  V_l=10000
5  V_ph=V_l/sqrt(3)
6  VA=10*10^6
7  I_FL=VA/(V_l*sqrt(3)) //Current at full laod
8  IX_s=(20/100)*V_ph //product of I and X_s
9
10 X_s=IX_s/I_FL
11 N_s=1500
12 f=50
13 P=120*f/N_s //poles
14
15 delta_dash_mech=%pi/180 //phase displacement in
    degree mechanical
16 delta_dash_elec=delta_dash_mech*(P/2)  //P/2 is pole
    pairs(and not poles)
17 E=V_ph //since alternator is on no-load
18 P_SY=delta_dash_elec*E^2/X_s  //Synchronous Power
19 P_SY_3ph=P_SY*3 //For 3 phases
20
21 printf('Synchronising Power of armature is %.3f kW.\
    nSynchronising Power for 3 phase is %.3f kW',P_SY
```

```
*10^-3,P_SY_3ph*10^-3)
```

Scilab code Exa 6.5 CALCULATE SYNCHRONISING POWER AND TORQUE AT NO LOAD AND FULL LOAD

```
clc,clear
printf('Example 6.5\n\n')
//note that a new function p2z has been defined
   below for direct representation of complex
   numbers in polar form
function [FUN] = p2z(RRRR,Theeeta)
 FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
endfunction

V_L=6.6*10^3
V_ph=V_L/sqrt(3)
VA=3*10^6
I_FL=VA/(V_L*sqrt(3)) //full load current
P=8,f=50 //poles and frequency

X_s=complex(0,2.9)//X_s=2.9
delta_dash_mech=%pi/180
delta_dash_elec=delta_dash_mech*(P/2)  //P/2 is pole
    pairs(and not poles)

//part(i)
E=V_ph
P_SY=delta_dash_elec*E^2/abs(X_s)  //Synchronous
   Power per phase
P_SY_3ph=P_SY*3 //For 3 phases
printf('(i)   Synchronising power at no load is %.3f
    kW',P_SY*10^-3)
printf('\n   Total synchronising power at no load is
    %.2f kW\n',P_SY_3ph*10^-3)

```

```
25 N_s=120*f/P //in rpm
26 n_s=(N_s)/60 //in rps
27 T_SY=P_SY_3ph/(2*%pi*n_s)
28 printf('\nSynchronous torque per mechanical degree
      of phase displacement is %.2f * 10^3 N-m',T_SY
      *10^-3)
29
30 //part(ii)
31 phi=acosd(0.85)
32 I=p2z(I_FL,0)
33 V=p2z(V_ph,phi)
34
35 E=V+I*X_s
36 //E leads I by phasemag(E). V leads I by phasemag(V)
37
38 delta=(%pi/180)* (phasemag(E)-phasemag(V) ) //power
      angle in radians
39 P_SY2=abs(E)*abs(V)*cos(delta)*sin(delta_dash_elec)/
      abs(X_s)
40
41 P_SY_total_2=3*P_SY2
42 //n_s=T_SY/(P_SY/(2*%pi) )        //because T_SY=P_SY
      /(2*%pi*n_s)
43 printf('\n\n(ii)Total synchronising power is %.0f kW
      ',P_SY_total_2*10^-3)
44
45 T_SY2=P_SY_total_2/(2*%pi*n_s)
46 printf('\nSynchronising torque is %.2f * 10^3 N-m',
      T_SY2/1000)
```

Scilab code Exa 6.6 TO DETERMINE PERCENTAGE CHANGE IN INDUCED EMF REQUIRED TO BRING UNITY POWER FACTOR

```
1 clc,clear
2 printf('Example 6.6\n\n')
```

```
3 //note that a new function p2z has been defined
     below for direct representation of complex
     numbers in polar form
4 function [FUN] = p2z(RRRR,Theeeta)
5  FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
6 endfunction
7
8 V_l=10*10^3
9 V_ph=V_l/sqrt(3)
10 R_a=0.4
11 Z=complex(R_a,6)
12 I_a=p2z(300,-acosd(0.8))
13 E=V_ph+I_a*Z
14
15 phi=acos(0.8)
16 alternator_op_ph=V_ph*abs(I_a)*cos(phi) //Power
     delivered to infinite bus per phase
17
18 //Power deliered to the altrernator = Power
     delivewred to bus bar +  I^2*R losses in armature
19 alternator_power= alternator_op_ph+ abs(I_a)^2*R_a
20
21 //this power developed remains constant.change pf to
      1 and calculate corresponding armature current
22 //alternator_power=V_ph*I_a1*cos(phi1)+I_a1^2*0.4
23 //solve the quadratic equation  0.4 I_a1^2+5773.50
     I_a1- 1421640 =0
24 I_a1=(-1*V_ph+sqrt(V_ph^2-4*R_a*-1*alternator_power)
     )/(2*R_a)
25
26 //also as follows
27 E1=V_ph+I_a1*Z
28 decrease=100*(abs(E)-abs(E1))/abs(E)
29 printf('Percentage decrease in induced e.m.f is %.1f
      percent',decrease)
```

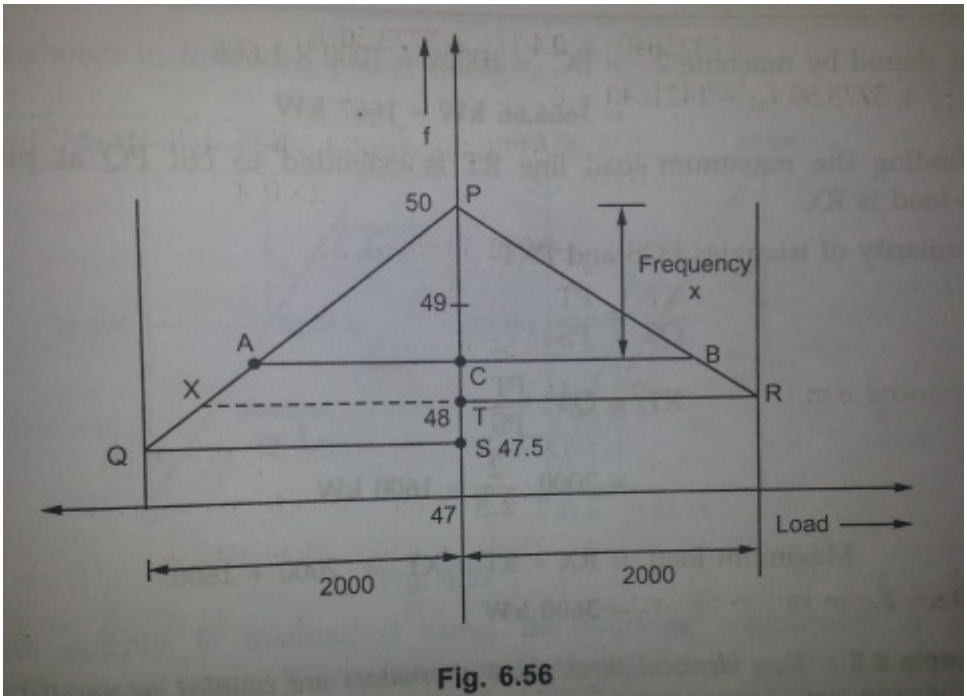


Figure 6.1: TO DETERMINE LOAD SHARING AND UPF MAXIMUM LOAD

Scilab code Exa 6.7 TO DETERMINE LOAD SHARING AND UPF MAXIMUM LOAD

```
1 clc,clear
2 printf('Example 6.7\n\n')
3
4 //Line PQ for Altermnator 1, and PR for alternaator
    2.AB is at frequency x from P where total load is
     3000 kW
5 QC=2000,PS=2.5,//PC=x
```

```
6  TR=2000,PT=2
7
8  //using similarity of triangles PAC and PQS
9  AC_by_PC=(QC/PS)// because (AC/QC)=(PC/PS)
10 //using similarity of triangles PCB and PTR
11 CB_by_PC=(TR/PT) // because (CB/TR)=(PC/PT)
12
13 AC_by_x=AC_by_PC    //which implies AC=12.5*x
14 CB_by_x=CB_by_PC   //which implies CB=16.67*x
15
16 AC_plus_CB=3000  //total load at the frequency at P
     is 30 kW
17 x= AC_plus_CB/(AC_by_x + CB_by_x)
18 AC=AC_by_x * x
19 CB=CB_by_x * x
20 frequency=50-x
21 printf('Loads shared by alternator 1 and 2 are %.2f
     kW and %.2f kW respectively',AC,CB)
22
23 //construction for max load: RT is extended to cut
     PQ at X.
24 QS=2000,RT=2000 //see figure
25 XT=QS*(PT/PS)
26 RX=RT+XT  //maximum load
27
28 printf('\nMaximum load is %.0f kW',RX)
```

Scilab code Exa 6.8 TO DETERMINE ARMATURE CURRENT OF ALTERNATOR 2 AND PF OF EACH ALTERNATORS

```
1 clc,clear
2 printf('Example 6.8\n\n')
3
```

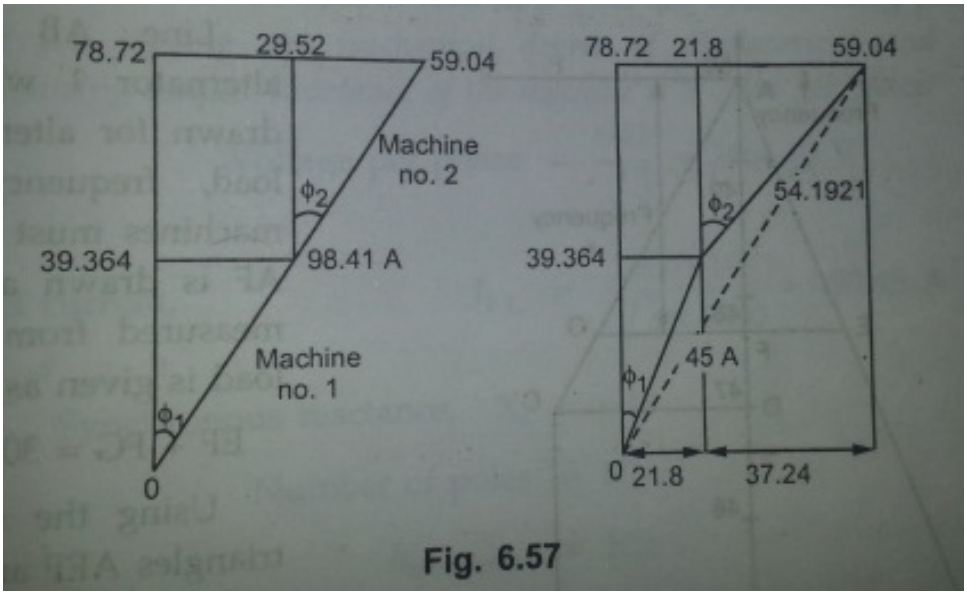


Figure 6.2: TO DETERMINE ARMATURE CURRENT OF ALTERNATOR 2 AND PF OF EACH ALTERNATORS

```
4  P_out=1500*10^3
5  V_L=11000
6  phi=acos(0.8)
7  I_L=P_out/(sqrt(3)*V_L*cos(phi))
8
9  I_L_actv=I_L*cos(phi)    //wattful or active
       component of current
10 I_L_reactive=I_L*sin(phi) //wattless or reactive
       component of current
11
12 I_each=I_L/2  //in identical conditions
13 I_arm1=45 //given
14 I_1_reactive=sqrt(I_arm1^2-39.364^2 )  //from the
       power triangle
15 I_2_reactive=59.046-21.80
16 I_a_2=sqrt( 39.364^2 + I_2_reactive^2 )  //required
       armature current of 2nd alternator
17 printf('Required armature current of second
```

```
    alternator is %.4f A\n',I_a_2)
//power factors of 2 machines
cos_phi1=39.364/45
cos_phi2=39.364/54.1921

printf('Power factors are %.4f lagging and %.4f
    lagging',cos_phi1,cos_phi2)
```

Scilab code Exa 6.9 TO DETERMINE LOAD ON EACH MACHINE

```
clc,clear
printf('Example 6.9\n\n')

//Line AB for Altermnator 1, and AC for alternator
    2.AF is at frequency x measured from A where
    total load is 3000 kW
BO=2000,AO=5//AF=x
DC=2000,AD=3,//AF=x

//using similarity of triangles AEF and ABO
EF_by_AF=(BO/AO)// because (EF/BO)=(AF/AO)
//using similarity of triangles AFG and ADC
FG_by_AF=(DC/AD) //because (FG/DC)=(AF/AD)

EF_by_x=EF_by_AF    //which implies EF=400*x
FG_by_x=FG_by_AF    //which implies FG=666.67*x

EF_plus_FG=3000  //total load at the frequency at P
    is 3000 kW
x= EF_plus_FG/(EF_by_x + FG_by_x)
EF=(BO/AO)*x
FG=(DC/AD)*x

```

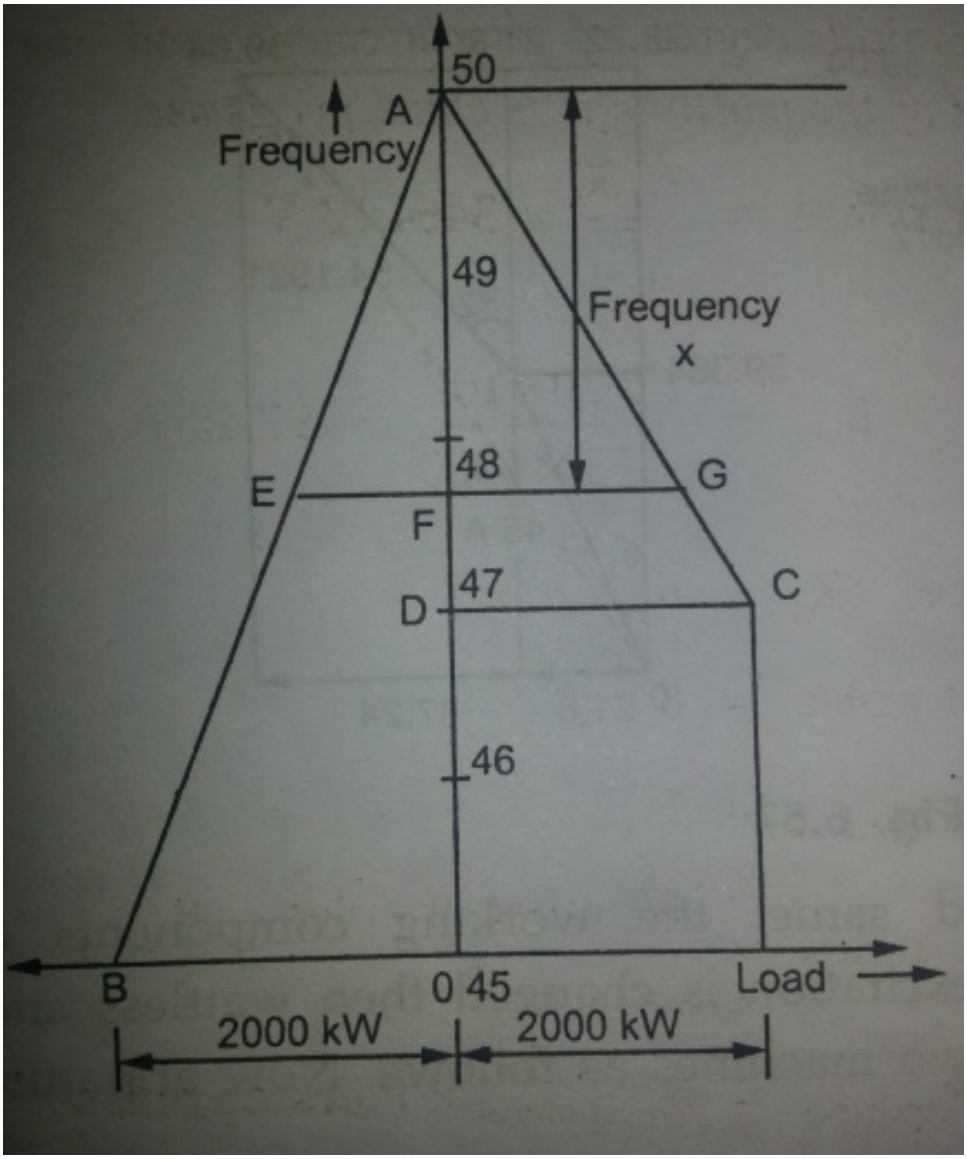


Figure 6.3: TO DETERMINE LOAD ON EACH MACHINE

```
21 printf('Loads shared by machine 1 and 2 are %.0f kW
     and %.0f kW respectively',EF,FG)
```

Scilab code Exa 6.10 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMNT AND CORRESPONDING SYNCHRONISING TORQUE

```
1  clc,clear
2  printf('Example 6.10\n\n')
3  //note that a new function p2z has been defined
     below for direct representation of complex
     numbers in polar form
4  function [FUN] = p2z(RRRR,Theeeta)
5   FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
6  endfunction
7
8  V_l=6000
9  V_ph=V_l/sqrt(3)
10 VA=2000*10^3
11 I_FL=VA/(V_l*sqrt(3))
12 X_s=complex(0,6) //synchronous reactance
13 P=8
14 f=50
15
16 delta_mech=%pi/180 //phase displacemant in degree
     mechanical
17 //phase displacemant in degree electrical
18 delta_elec=delta_mech*(P/2)  //P/2 is pole pairs(and
      not poles)
19
20 phi=acosd(0.8)
21 V=p2z(V_ph,phi)
22 E=V+I_FL*X_s
23 //E leads I by phasemag(E). V leads I by phasemag(V)
24
```

```
25 delta=(%pi/180)* (phasemag(E)-phasemag(V) ) //power
     angle in radians
26 P_SY=abs(E)*abs(V)*cos(delta)*sin(delta_elec)/abs(
     X_s) //synchronising power
27 P_SY_total=3*P_SY //totla synchronising power
28 printf('Total synchronising power is %.3f kW',10^-3*
     P_SY_total)
29
30 N_s=120*f/P //in rpm
31 n_s=(N_s)/60 //in rps
32 T_SY=P_SY_total/(2*%pi*n_s)
33 printf('\nSynchronising torque is %.0f N-m',T_SY)
```

Scilab code Exa 6.11 TO CALCULATE SYNCHRONISING POWER AND SYNCHRONISING TORQUE

```
1 clc,clear
2 printf('Example 6.11\n\n')
3
4 //note that a new function p2z has been defined
     below for direct representation of complex
     numbers in polar form
5 function [FUN] = p2z(RRRR,Theeeta)
6  FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
7 endfunction
8
9 V_l=3300
10 V_ph=V_l/sqrt(3)
11 VA=3*10^6
12 I_FL=VA/(V_l*sqrt(3))
13 IX_s=(25/100)*V_ph //product of I and X_s
14 X_s=complex(0,IX_s/I_FL) //synchronous reactance
15 N_s=1000 //in rpm
16 P=6
17 f=50
```

```
18
19 delta_dash_mech=%pi/180
20 delta_dash_elec=delta_dash_mech*(P/2)  //P/2 is pole
       pairs(and not poles)
21
22 I=I_FL
23 phi=acosd(0.8)
24 V=p2z(V_ph,phi)
25 E=V+I*X_s
26 //E leads I by phasemag(E). V leads I by phasemag(V)
27
28 delta=(%pi/180)* (phasemag(E)-phasemag(V) ) //power
      angle in radians
29 P_SY=abs(E)*abs(V)*cos(delta)*sin(delta_dash_elec)/
      abs(X_s) //Synchronising power per phase
30 printf('Synchronising power is %.3f kW',10^-3*P_SY)
31 P_SY_total=3*P_SY //Total synchronising power
32
33 N_s=120*f/P //in rpm
34 n_s=(N_s)/60 //in rps
35 T_SY=P_SY_total/(2*%pi*n_s)
36 printf('\nSynchronising torque is %.0f N-m',T_SY)
37
38 printf('\n\nAnswer mismatches due to approximation')
```

Scilab code Exa 6.12 TO CALCULATE SYNCHRONISING POWER AND CORRESPONDING SYNCHRONISING TORQUE

```
1 clc,clear
2 printf('Example 6.12\n\n')
3
4 V_l=3300
5 V_ph=V_l/sqrt(3)
6 VA=3*10^6
7 I_FL=VA/(V_l*sqrt(3))
```

```
IX_s=(20/100)*V_ph //product of I and X_s
X_s=complex(0,IX_s/I_FL) //synchronous reactance
N_s=1000
P=6
f=50

delta_dash_mech=%pi/180 //phase displacement in
   degree mechanical
//phase displacement in degree electrical
delta_dash_elec=delta_dash_mech*(P/2)  //P/2 is pole
    pairs(and not poles)

E=V_ph
Z_s=X_s //since R=0
P_SY=abs(E)*abs(V_ph)*delta_dash_elec/abs(Z_s) //
   Synchronising power per phase
printf('Synchronising power is %.3f kW',10^-3*P_SY)
P_SY_total=3*P_SY //Total synchronising power
printf('\n3 phase synchronising power is %.3f kW'
   ,10^-3*P_SY_total)

N_s=120*f/P //in rpm
n_s=(N_s)/60 //in rps
T_SY=P_SY_total/(2*%pi*n_s)
printf('\nSynchronising torque is %.0f N-m',T_SY)
```

Scilab code Exa 6.13 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT

```
clc,clear
printf('Example 6.13\n\n')

V_L=11*10^3
```

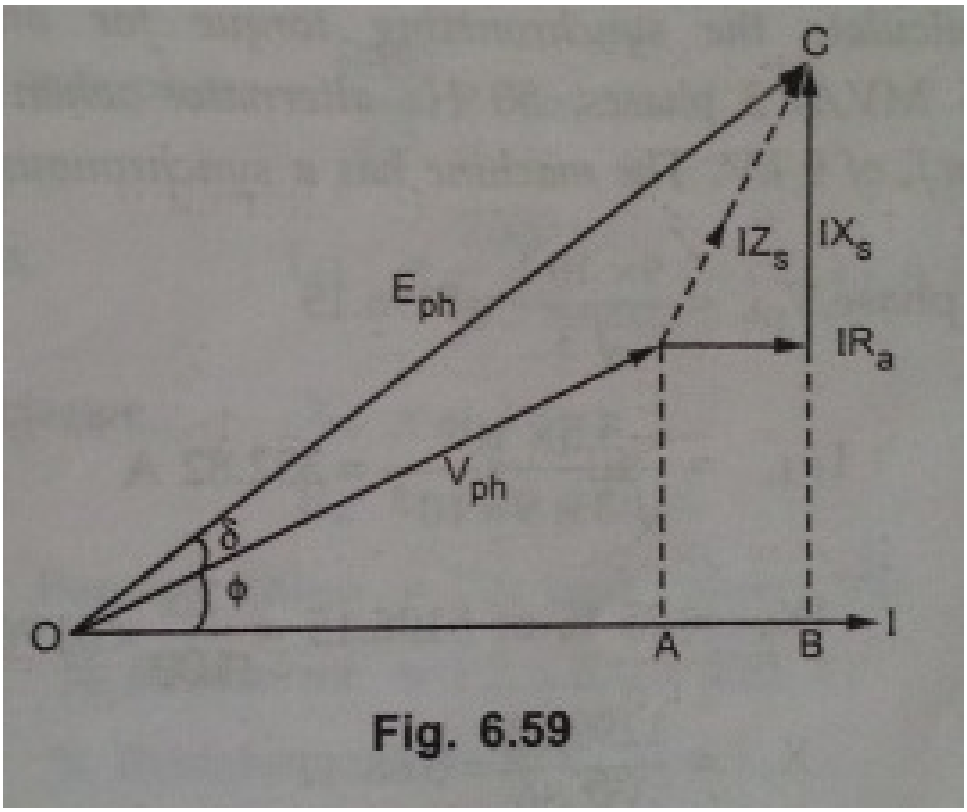


Figure 6.4: TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT

```
V_ph=V_L/sqrt(3)
VA=700*10^3
I_FL=VA/(sqrt(3)*V_L) //full load current
IR_a=(1.5/100)*V_ph //product of I and R_a
R_a=IR_a/I_FL
IX_s=(14/100)*V_ph // product of I and X_s
X_s=IX_s/I_FL //synchronous reactance

//at full load and 0.8 pf
I=I_FL
phi=acos(0.8)
V_ph=complex(V_ph*cos(phi),V_ph*sin(phi))  //just
   introduced the angle
E_ph=sqrt(    (abs(V_ph)*cos(phi)+ IR_a)^2+   (abs(
   V_ph)*sin(phi)+ IX_s)^2     )

Poles=4,f=50 //poles and frequency
delta=asin( (abs(V_ph)*sin(phi)+IX_s)/E_ph)  -phi
delta_dash_mech=(%pi/180) //displacement in degree
   mechanical
//displacement in degree electrical
delta_dash_elec=delta_dash_mech*(Poles/2)
P_SY=abs(E_ph)*abs(V_ph)*cos(delta)*sin(
   delta_dash_elec)/X_s   //synchronising power per
   phase
P_SY_total=3*P_SY  //total synchronising power

ns=120*f/(60*Poles)  //in r.p.s
T_SY=P_SY_total/(2*%pi*ns)   //Synchronising torque
printf('Synchronising power is %.2fkW\n',P_SY_total
   /1000)
printf('Synchronising torque is %.2f N-m',T_SY)
```

Scilab code Exa 6.14 TO CALCULATE SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT

```
clc,clear
printf('Example 6.14\n\n')

V_l=9*10^3
V_ph=V_l/sqrt(3)
VA=5.5*10^6
I_FL=VA/(V_l*sqrt(3))
IX_s=(25/100)*V_ph //product of I and X_s
X_s=complex(0,IX_s/I_FL) //synchronous reactance
N_s=1500 //in rpm
n_s=N_s/60 //in rps
f=50,P=120*f/N_s //frequency and pole

delta_dash_mech=%pi/180 //displacemnt in degree
   mechanical
//displacemnt in degree electrical
delta_dash_elec=delta_dash_mech*(P/2)  //P/2 is pole
    pairs(and not poles)

E=V_ph
P_SY=abs(E)*abs(V_ph)*delta_dash_elec/abs(X_s) //
   Synchronising power per phase
P_SY_total=3*P_SY //Total synchronising power

T_SY=P_SY_total/(2*%pi*n_s)
printf('\nSynchronising torque is %.2f N-m',T_SY)
printf('\nAnswer mismatches due to approximation')
```

Scilab code Exa 6.15 TO CALCULATE SYNCHRONISING POWER SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT

```
clc,clear
```

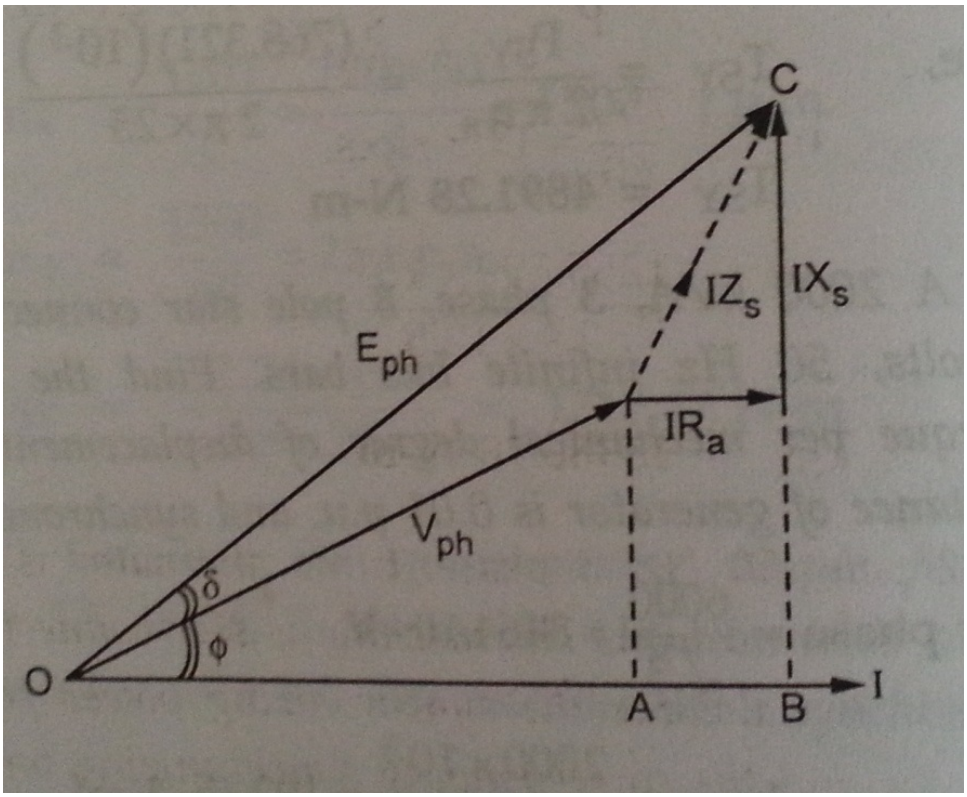


Figure 6.5: TO CALCULATE SYNCHRONISING POWER SYNCHRONISING TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT

```
2  printf('Example 6.15\n\n')
3
4  V_L=6*10^3
5  V_ph=V_L/sqrt(3)
6  VA=2000*10^3
7  I_FL=VA/(sqrt(3)*V_L)  ,I=I_FL
8
9  X_s=1.2,R_a=0.01 //both per unit
10 IR_a=(1/100)*V_ph //product of I and R_a
11 R_a=IR_a/I_FL
12 IX_s=(120/100)*V_ph //product of I and X_s
13 //IX_s=(12/100)*V_ph   // this is the mistake made
     in the textbook
14 X_s=IX_s/I_FL
15
16 //at full load and 0.8 pf
17 phi=acos(0.8)
18 //V_ph=complex(V_ph*cos(phi),V_ph*sin(phi))  //just
     introduced the angle
19 E_ph=sqrt(    (abs(V_ph)*cos(phi)+ IR_a)^2+   (abs(
     V_ph)*sin(phi)+ IX_s)^2     )
20 Poles=8,f=50
21
22 delta=asin( (abs(V_ph)*sin(phi)+IX_s)/E_ph)  -phi
23 delta_dash_mech=(%pi/180) //displacemnt in degree
     mechanical
24 //displacemnt in degree electrical
25 delta_dash_elec=delta_dash_mech*(Poles/2)
26 P_SY=abs(E_ph)*abs(V_ph)*cos(delta)*sin(
     delta_dash_elec)/X_s   //synchronising power per
     phase
27 P_SY_total=3*P_SY  //total synchronising power
28
29 ns=120*f/(60*Poles)  //in r.p.s
30 T_SY=P_SY_total/(2*%pi*ns)   //Synchronising torque
31
32 printf('Synchronising power is %.2f kW\n',P_SY_total
     /1000)
```

```
33 printf('Synchronising torque is %.2f N-m',T_SY)
34
35 printf('\n\nNote that answer obtained doesnt match
   with textbook due to the following reasons: \n(i)
   IX_s is considered wrong in textbook.\nIt should
   have been 4156.92(instead of 415.692) \nTo
   verify this use commented statement of IX_s (line
    13)and notice that it matches with textbook ans
   then' )
```

Scilab code Exa 6.16 TO CALCULATE SYNCHRONIZING POWER PER MECHANICAL DEGREE OF DISPLACEMENT AND CORRESPONDING SYNCHRONIZING TORQUE

```
1 clc,clear
2 printf('Example 6.16\n\n')
3
4 E=11*10^3/sqrt(3)
5 I_sc=1000,Pole=2,f=50
6 delta_dash_mech=1*%pi/180 //displacemnt in degree
   mechanical
7 //displacemnt in degree electrical
8 delta_dash_elec=delta_dash_mech*(Pole/2)
9 P_SY=E*I_sc*delta_dash_mech //synchronising power
   per phase
10 P_SY_total=P_SY*3 //total synchronising power
11
12 ns=120*f/(60*Pole)  //in r.p.s
13 T_SY=P_SY_total/(2*%pi*ns)   //Synchronising torque
14
15 printf('Synchronising power is %.2f kW\n',P_SY_total
   /1000)
16 printf('Synchronising torque is %.2f N-m',T_SY)
```

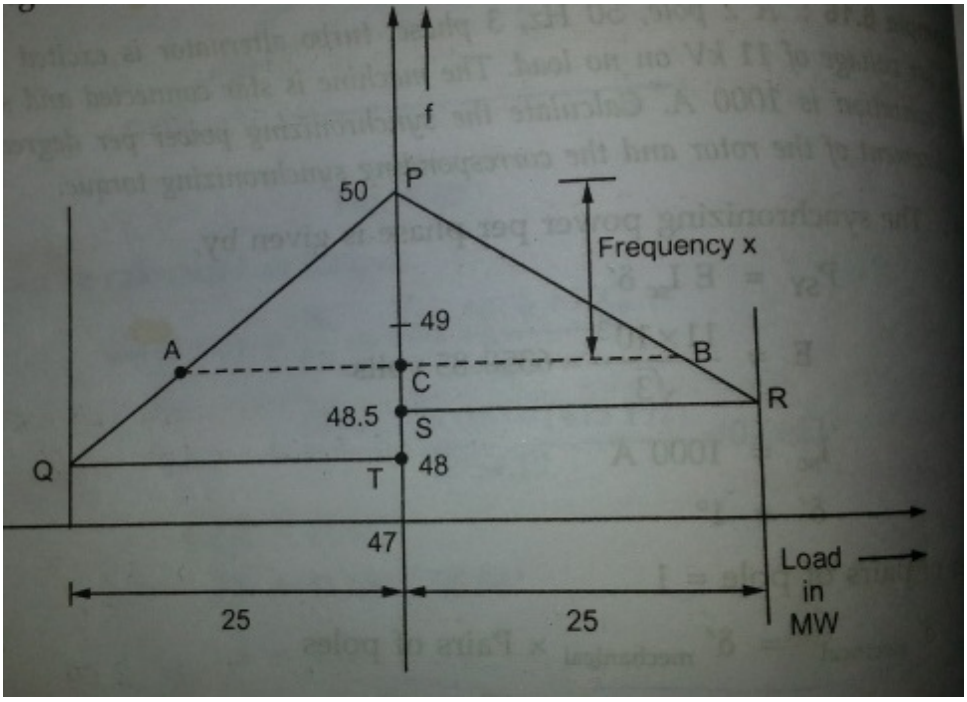


Figure 6.6: DETERMINE THE LOAD SHARED BY EACH OF THE 2 MACHINES

Scilab code Exa 6.17 DETERMINE THE LOAD SHARED BY EACH OF THE 2 MACHINES

```
1 clc,clear
2 printf('Example 6.17\n\n')
3 //Line PQ for Altermnator 1, and PR for alternaator
    2.AB is at frequency x from P where total load is
     30 MW
4 QT=25,PT=2,//PC=x
5 SR=25,PS=1.5
```

```
6
7 //using similarity of triangles PAC and PQT
8 AC_by_PC=(QT/PT)// because (AC/QT)=(PC/PT)
9 //using similarity of triangles PCB and PSR
10 CB_by_PC=(SR/PS)
11
12 AC_by_x=AC_by_PC    //which implies AC=12.5*x
13 CB_by_x=CB_by_PC   //which implies CB=16.67*x
14
15 AC_plus_CB=30  //total load at the frequency at P is
       30 MW
16 x= AC_plus_CB/(AC_by_x + CB_by_x)
17 AC=12.5*x
18 CB=16.67*x
19 frequency=50-x
20 printf('Loads shared by alternator 1 and 2 are %.2f
     MW and %.2f MW respectively',AC,CB)
```

Scilab code Exa 6.18 TO DETERMINE THE EXCITATION OF 2ND ALTERNATORS

```
1 clc,clear
2 printf('Example 6.18\n\n')
3 //note that a new function p2z has been defined
     below for direct representation of complex
     numbers in polar form
4 function [FUN] = p2z(RRRR,Theeeta)
5  FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
6 endfunction
7
8 load_total=1600*10^3
9 pf=1/sqrt(2)  //lag
10 V_L=6600
```

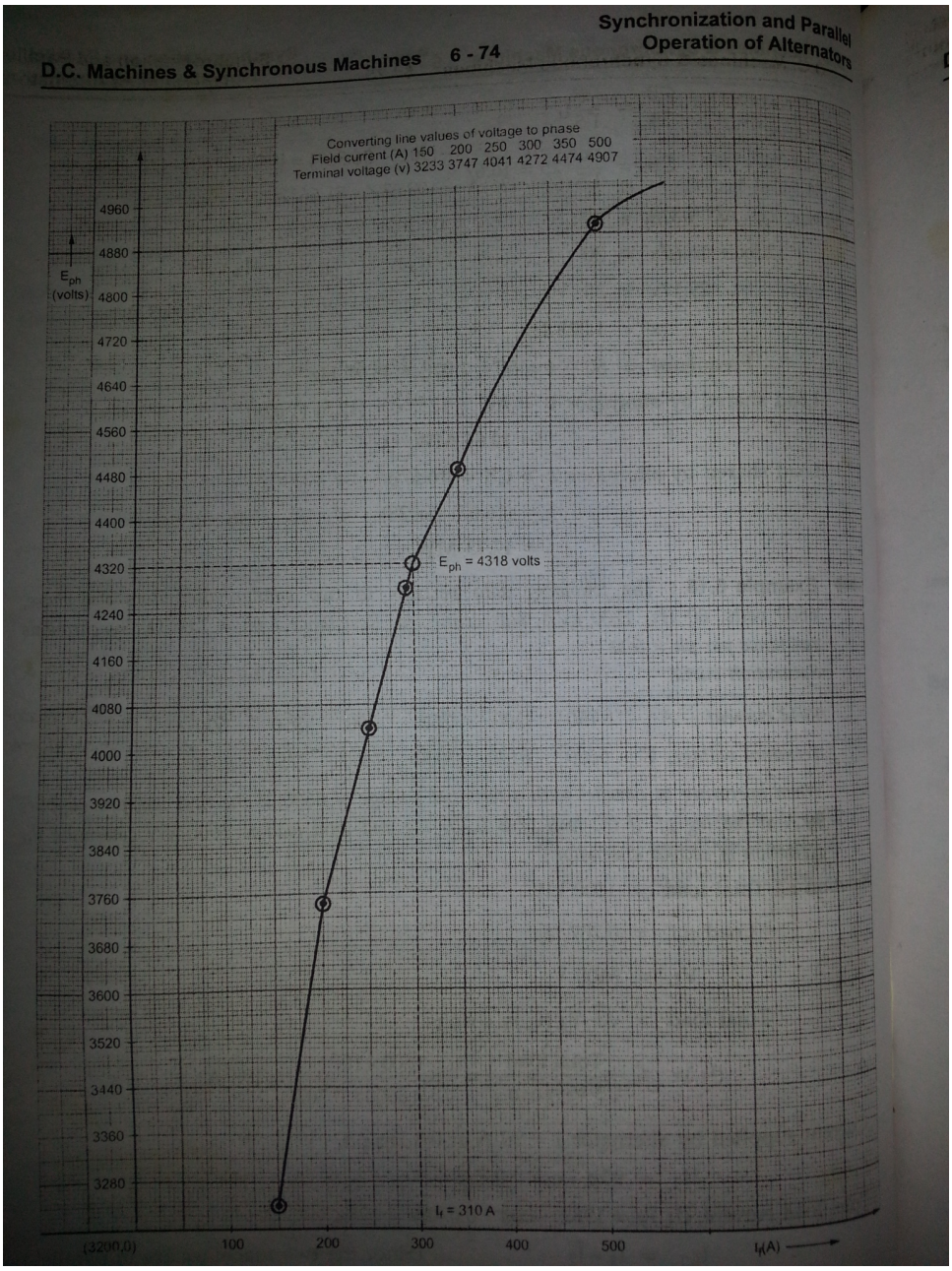


Figure 6.7: TO DETERMINE THE EXCITATION OF 2ND ALTERNATORS

```
I_L=p2z(load_total/(sqrt(3)*V_L*pf),-1*acosd(pf))
I_1=p2z(90,-1*acosd(0.8))
I_2=I_L-I_1
phi=abs(phasemag(I_2))
I_a=abs(I_2)
R_a=1.05,X_s=5 //resistance and synchronous
   reactance per phase
V_ph=V_L/sqrt(3)
E_ph=sqrt(  (V_ph*cos(phi)+I_a*R_a )^2 + ( V_ph*sin(
   phi)+I_a*X_s )^2        )
E_line=sqrt(3)*E_ph

printf('Excitation of second alternator is %.2f V ',
   E_line)
printf('\n The corresponding field current from the
   graph is about 310 A\n\n')
printf('Note: The answer obtained will differ from
   textbook answer because of higher degree \nof
   accuracy while storing I_2 and the improper
   rounding off of I_2 in the textbook')
```

Scilab code Exa 6.19 TO DETERMINE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT UNDER NOLOAD

```
clc,clear
printf('Example 6.19\n\n')

V_L=10*10^3
V_ph=V_L/sqrt(3)
VA=5*10^6
I_FL=VA/(sqrt(3)*V_L) //full-load current
IX_s=(20/100)*V_ph //product of I and X_s
X_s=IX_s/I_FL //synchronous reactance
P=4
delta_dash_mech=1*(%pi/180) //displacement in degree
```

```
    mechanical
//displacement in degree electrical
delta_dash_elec=delta_dash_mech*(P/2)
E=V_ph  //at no load
P_SY= delta_dash_elec*E^2/X_s  //synchronising power
    per phase
P_SY_total=P_SY*3  //Total synchronising power

printf('Synchronising power per phase is %.2fkW\
   nTotal synchronising power is %.2fkW ',P_SY/1000,
   P_SY_total/1000)
```

Scilab code Exa 6.20 TO FIND EMF AND POWER ANGLE

```
clc,clear
printf('Example 6.20\n\n')

Power_total=1.414 //per unit
V_L=1 //per unit
phi_t=acos(0.707)
I_L_T=Power_total/(sqrt(3)*V_L*cos(phi_t))  //Total
   current
//Current supplied by each alternator
I_1=I_L_T/2
I_2=I_1
V_ph=V_L/sqrt(3)

phi=acos(0.707)
R_a=0,X_s=0.6 //resistacne and synchronous reactance
E_ph=sqrt(  (V_ph*cos(phi)+ I_1*R_a)^2 + (V_ph*sin(
   phi)+I_1*X_s)^2          )
delta= atan((I_1*X_s+V_ph*sin(phi)) / (V_ph*cos(phi)
   )) - phi    //power angle
```

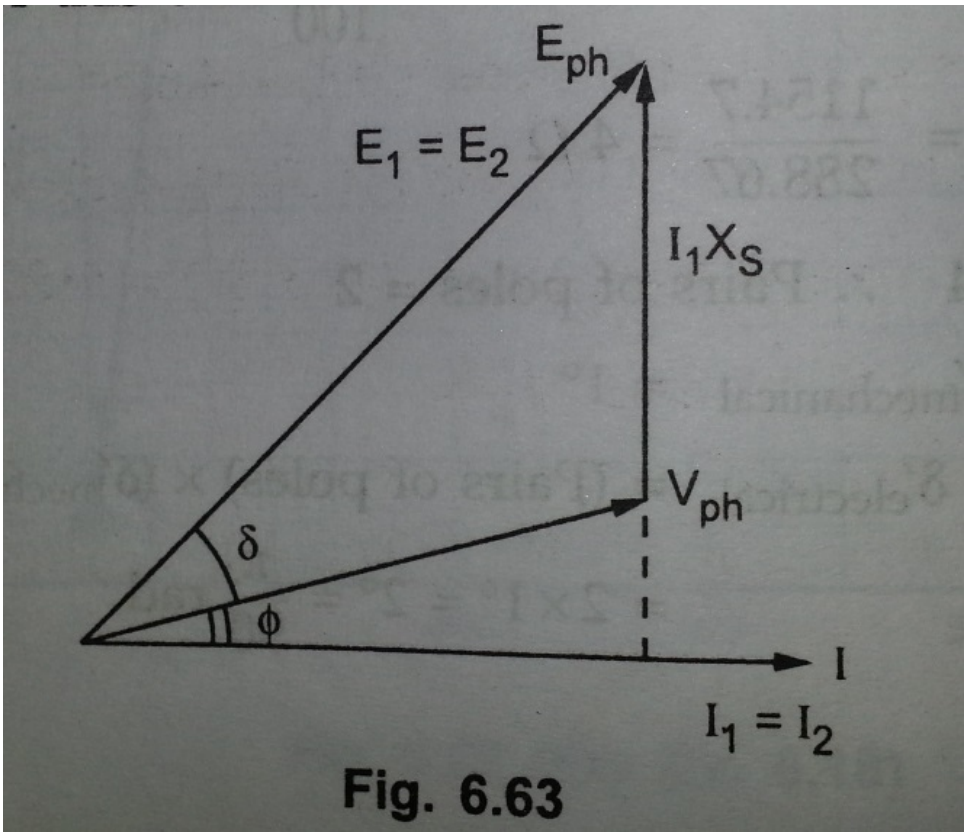


Figure 6.8: TO FIND EMF AND POWER ANGLE

```
17
18 printf('EMF is %.4f p.u. and power angle is %.2f
      degrees ',E_ph,delta*(180/%pi))
19 printf('\n\nFollowing assumptions were made :\n')
20 printf('1.Terminal or bus bar voltage at ppoint of
      connection is constant\n')
21 printf('2.The alternators are identical and are
      initially equally excited\n')
22 printf('3.The power supplied by prime movers is
      adjusted so that each machine carries half the
      load represented by external impedance Z=R+ j 2
      pifL , where R and L are constant\n')
23 printf('4.The stator resistance is negligible')
```

Scilab code Exa 6.21 TO FIND THE EXCITATION EMF

```
1 clc,clear
2 printf('Example 6.21\n\n')
3
4 V_l=480
5 X_d=0.1,X_q=0.075,R_a=0 //armature resistance and
      synchronous reactance of direct,quadrature axis
6 I_l=1200
7 I_ph=I_l/sqrt(3)
8 V_ph=V_l
9 V_t=V_l,I_a=I_ph
10 phi=acos(0.8)
11 psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
      R_a)            )
12 delta=psi-phi
13
14 I_d=I_a*sin(psi)
15 I_q=I_a*cos(psi)
16 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
17
```

```
18 printf('Excitation e.m.f is %.2f V ',E_f)
```

Scilab code Exa 6.22 TO DETERMINE REGULATION AND EXCITATION EMF REQUIRED TO MAINTAIN CERTAIN TERMINAL VOLTAGE

```
1  clc,clear
2  printf('Example 6.22\n\n')
3
4  VA=3.5*10^6
5  P=32          //Poles
6  Power=2.5*10^6 //In watts
7  V_l=6.6*10^3
8  phi=acos(0.8)
9  I_l=Power/(V_l*cos(phi)*sqrt(3))
10 X_d=9.6,X_q=6,R_a=0 //armature resistance and
     synchronous reactance of direct,quadrature axis
11
12 V_t=V_l/sqrt(3)
13 psi=atan(  (V_t*sin(phi)+I_l*X_q)/(V_t*cos(phi)+I_l*
     R_a)             )
14 delta=psi-phi
15 I_s=I_l
16 I_d=I_s*sin(psi)
17 I_q=I_s*cos(psi)
18 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
19
20 regulation=100*(E_f-V_t)/V_t
21 printf('percentage regulation is %.2f percent',
     regulation)
22 printf('\nExcitation emf= %.0f V',E_f)
```

Scilab code Exa 6.23 TO CALCULATE PERCENTAGE REGULATION OF THE MACHINE

```
clc,clear
printf('Example 6.23\n\n')

X_d=7.6,X_q=4.5,R_a=0.15 //armature resistance and
   synchronous reactance of direct,quadrature
   axisV_l=13.8*10^3
V_l=13.8*10^3
V_t=V_l/sqrt(3)
phi=acos(0.8)
VA=25*10^6
I_a=VA/(sqrt(3)*V_l)
psi=atan(  (V_t*cos(phi)+I_a*X_q)/(V_t*sin(phi)+I_a*
   R_a)            )

delta=psi-phi
I_s=I_a
I_d=I_s*sin(psi)
I_q=I_s*cos(psi)

E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
regulation=100*(E_f-V_t)/V_t

printf('percentage regulation is %.2f percent',
   regulation)
```

Scilab code Exa 6.24 TO CALCULATE PERCENTAGE VOLTAGE REGULATION AT A CERTAIN PF

```
clc,clear
printf('Example 6.24\n\n')

X_d=1,X_q=0.6,R_a=0 //armature resistance and
```

```
    synchronous reactance of direct,quadrature axis
phi=acos(0.8) //lag
V_t=1
I_a=1 //full load
psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
    R_a)            )

delta=psi-phi
I_s=I_a
I_d=I_a*sin(psi)
I_q=I_a*cos(psi)

E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
regulation=100*(E_f-V_t)/V_t
printf('percentage regulation is %.2f percent',
    regulation)
```

Scilab code Exa 6.25 TO DETERMINE LOAD ANGLE AND COMPONENTS OF ARMATURE CURRENT

```
clc,clear
printf('Example 6.25\n\n')

I_a=10
phi=20 //lag and degrees
V_t=400
X_d=10,X_q=6.5,R_a=0 //armature resistance and
    synchronous reactance of direct,quadrature axis

psi=atand(  (V_t*sind(phi)+I_a*X_q)/(V_t*cosd(phi)+
    I_a*R_a)            )
delta=psi-phi
I_d=I_a*sind(psi)
I_q=I_a*cosd(psi)

```

```
14 printf('Load angle is %.2f degrees \n',delta)
15 printf('I_d and I_q are %.4f A and %.4f A
      respectively ',I_d,I_q  )
```

Scilab code Exa 6.26 TO COMPUTE PERCENTAGE REGULATION AT DIFFERENT POWER FACTOR

```
1  clc,clear
2  printf('Example 6.26\n\n')
3
4  X_d=0.8,X_q=0.5,R_a=0.02 //armature resistance and
      synchronous reactance of direct,quadrature axis
5
6  //case(i)  lag
7  phi=acos(0.8)
8  V_t=1
9  I_a=1//full-load
10 psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
      R_a)            )
11 delta=psi-phi
12
13 I_d=I_a*sin(psi)
14 I_q=I_a*cos(psi)
15
16 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
17 regulation=100*(E_f-V_t)/V_t
18 printf('percentage regulation at 0.8 pf lag is %.2f
      percent',regulation)
19
20 //case(ii)  lead
21 phi2=-1*acos(0.8) //minus sign because of leading pf
22 psi2=atan(  (V_t*sin(phi2)+I_a*X_q)/(V_t*cos(phi2)+
      I_a*R_a)            )
23 delta2=psi2-phi2
24
```

```
I_d2=I_a*sin(psi2)
I_q2=I_a*cos(psi2)

E_f2=V_t*cos(delta2)+I_d2*X_d+I_q2*R_a
regulation2=100*(E_f2-V_t)/V_t
printf('\npercentage regulation at 0.8 pf lead is %
    .2f percent',regulation2)
```

Scilab code Exa 6.27 TO CALCULATE THE OUTPUT POWER FACTOR OF SECOND ALTERNATOR

```
clc,clear
printf('Example 6.27\n\n')

kW=[800,500,1000,600]
cosphi=[1,0.9,0.8,0.9]
tanphi=tan(acos(cosphi))
kVAR=kW.*tanphi

kW_total=kW(1)+kW(2)+kW(3)+kW(4)
kVAR_total=kVAR(1)+kVAR(2)+kVAR(3)+-1*kVAR(4) //4th
    case is leading

phi_c=atan(kVAR_total/kW_total)  //total power
    factor angle
phi_1=acos(0.95)//pf of machine 1
kW_1=1000 //active component of machine 1
kVAR_1=kW_1*tan(phi_1) //reactive component of
    machine 1
kW_2=kW_total - kW_1 //active component of machine 1
kVAR_2=kVAR_total-kVAR_1 //reactive component of
    machine 2

phi_2=atan(kVAR_2/kW_2)
pf_2=cos(phi_2) //power factor of machine 2
```

```
21
22 printf('Output of second alternator= %.0f kW',kW_2)
23 printf('\npower factor of machine 2 = %.2f and
   lagging',pf_2)
```

Scilab code Exa 6.28 TO CALCULATE THE POWER FACTOR OF SECOND MACHINE WORKING PARALLEL TO THE FIRST MACHINE

```
1  clc,clear
2  printf('Example 6.28\n\n')
3
4  kW=[250,300,150]
5  cosphi=[0.9,0.75,0.8] //all lagging
6  tanphi=tan(acos(cosphi))
7  kVAR=kW.*tanphi
8
9  kW_total=kW(1)+kW(2)+kW(3)
10 kVAR_total=kVAR(1)+kVAR(2)+kVAR(3)
11
12 phi_1=acos(0.8)//pf of machine 1
13 kW_1=100 //active component of machine 1
14 kVAR_1=kW_1*tan(phi_1) //reactive component of
   machine 1
15 kW_2=kW_total - kW_1 //active component of machine 1
16 kVAR_2=kVAR_total-kVAR_1 //reactive component of
   machine 2
17 phi_2=atan(kVAR_2/kW_2)
18 pf_2=cos(phi_2) //power factor of machine 2
19
20 printf('Output of second alternator= %.0f kW',kW_2)
21 printf('\npower factor of machine 2 = %.4f and
   lagging',pf_2)
```

Scilab code Exa 6.29 TO DETERMINE VOLTAGE REGULATION AND OPEN CIRCUIT POWER SUPPLY OF GENERATOR

```
clc,clear
printf('Example 6.29\n\n')

V_L=6.6*10^3
V_ph=V_L/sqrt(3)
V_t=V_ph
X_d=9.6,X_q=6,R_a=0 //armature resistance and
   synchronous reactance of direct,quadrature axis
VA=3.5*10^6
I_L=VA/(sqrt(3)*V_L)

P=2.5*10^6,phi=acos(0.8)
I_a=P/(sqrt(3)*V_L*cos(phi))
psi=atan(  (V_t*sin(phi)+ I_a*X_q)/(V_t*cos(phi)+
   I_a*R_a)     )

delta=psi-phi
I_d=I_a*sin(psi)
I_q=I_a*cos(phi)

E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
regulation=100*(E_f-V_t)/V_t
P_max=(V_ph^2/2)*((X_d-X_q)/(X_d*X_q))*(sin(2*delta)
   )

printf('percentage voltage regulation is %.2f
   percent',regulation)
printf('\nPower under open circuit is %.1f kW per
   phase',P_max/1000)
```

Scilab code Exa 6.30 TO CALCULATE SYNCHRONISING POWER AND TORQUE PER MECHANICAL DEGREE OF DISPLACEMENT

```
clc,clear
printf('Example 6.30\n\n')

V_L=3.3*10^3
V_ph=V_L/sqrt(3)
VA=3*10^6
I_FL=VA/(sqrt(3)*V_L)
IX_s=(25/100)*V_ph //product of I and X_s
X_s=complex(0,IX_s/I_FL)
N_s=1000 //in r.p.m

Poles=6,f=50
delta_dash_mech=(%pi/180) //displacement in degree mechanical
//displacement in degree electrical
delta_dash_elec=delta_dash_mech*(Poles/2)

I=I_FL,phi=acos(0.8)
V=complex(V_ph*cos(phi),V_ph*sin(phi))
E= V+ I*X_s

delta=(%pi/180)*phasemag(E)-phi    //E leads I by (%pi/180)*phasemag(E) and V leads I by phi radians
P_SY=abs(E)*abs(V_ph)*cos(delta)*sin(delta_dash_elec)/abs(X_s)    //synchronising power per phase
P_SY_total=3*P_SY  //total synchronising power

ns=120*f/(60*Poles)  //in r.p.m
T_SY=P_SY_total/(2*%pi*ns)    //Synchronising torque
printf('Synchronising power per phase is %.3f kW\n',P_SY/1000)
printf('Synchronising torque is %.0f N-m',T_SY)
printf('\n\nAnswer mismatches due to improper approximation')
```

Scilab code Exa 6.31 TO CALCULATE SYNCHRONIZING POWER PER MECHANICAL DEGREE OF DISPLACEMENT AND CORRESPONDING SYNCHRONISING TORQUE

```
clc,clear
printf('Example 6.31\n\n')

V_L=3.3*10^3
V_ph=V_L/sqrt(3)
VA=3*10^6
I_FL=VA/(sqrt(3)*V_L)
IX_s=(20/100)*V_ph //product of I and X_s
X_s=complex(0,IX_s/I_FL)
N_s=1000 //in r.p.m
Poles=6,f=50

delta_dash_mech=(%pi/180) //displacement in degree mechanical
//displacement in degree electrical
delta_dash_elec=delta_dash_mech*(Poles/2)

//E=V as the alternator is on no-load and X_s=Z_s
P_SY=abs(V_ph)^2*(delta_dash_elec)/abs(X_s)   //synchronising power per phase
P_SY_total=3*P_SY  //total synchronising power

ns=120*f/(60*Poles)  //in r.p.s
T_SY=P_SY_total/(2*%pi*ns)   //Synchronising torque
printf('Synchronising power per phase is %.3f kW\n',P_SY/1000)
printf('Total Synchronising power is %.3f kW',P_SY_total/1000)
printf('\nSynchronising torque is %.0f N-m',T_SY)
```

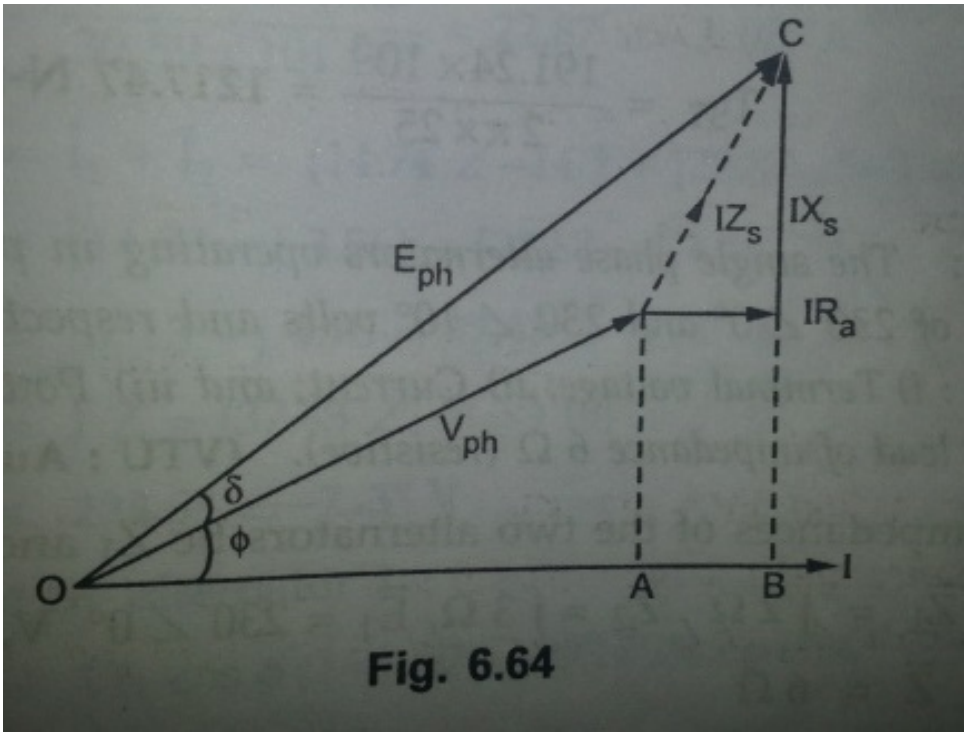


Figure 6.9: TO DETERMINE SYNCHRONOUS POWER PER MECHANICAL DEGREE OF DISPLACEMENT AT FULL LOAD

Scilab code Exa 6.32 TO DETERMINE SYNCHRONOUS POWER PER MECHANICAL DEGREE OF DISPLACEMENT AT FULL LOAD

```
clc,clear
printf('Example 6.32\n\n')

V_L=11*10^3
V_ph=V_L/sqrt(3)
VA=700*10^3
I_FL=VA/(sqrt(3)*V_L)
IX_s=(14/100)*V_ph //product of I and X_s
X_s=IX_s/I_FL
//X_s=complex(0,IX_s/I_FL)
IR_a=(1.5/100)*V_ph //product of I and R_a
R_a=IR_a/I_FL

I=I_FL,phi=acos(0.8)
V=complex(V_ph*cos(phi),V_ph*sin(phi))
E_ph=sqrt(  (V_ph*cos(phi)+IR_a)^2 +(V_ph*sin(phi)+
   IX_s)^2    )

delta=asin((V_ph*sin(phi)+IX_s)/E_ph)   -phi

Poles=4,f=50
delta_dash_mech=(%pi/180) //phase displacemnt in
   degree mechanical
delta_dash_elec=delta_dash_mech*(Poles/2)//phase
   displacemnt in degree electrical

P_SY=abs(V_ph)*abs(E_ph)*cos(delta)*sin(
   delta_dash_elec)/abs(X_s)   //synchronising power
    per phase
P_SY_total=3*P_SY  //total synchronising power

ns=120*f/(60*Poles)  //in r.p.s
T_SY=P_SY_total/(2*%pi*ns)   //Synchronising torque
printf('Synchronising power per phase is %.3f kW\n',
   P_SY/1000)
```

```
30 printf('Synchronising power is %.3f kW ; ',P_SY
   /1000)
31 printf('Total Synchronising power is %.3f kW',
   P_SY_total/1000)
32 printf('\nSynchronising torque is %.2f N-m',T_SY)
```

Scilab code Exa 6.33 TO DETERMINE CERTAIN CHARACTERISTICS OF TWO ALTERNATORS OPERATING IN PARALLEL

```
1  clc,clear
2  printf('Example 6.33\n\n')
3  //note that a new function p2z has been defined
   below for direct representation of complex
   numbers in polar form
4  function [FUN] = p2z(RRRR,Theeeta)
5   FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
6  endfunction
7
8  Z1=complex(0,2)
9  Z2=complex(0,3)
10 Z=6
11 E1=p2z(230,0)
12 E2=p2z(230,10)
13
14 I1=((E1-E2)*Z+E1*Z2)/(Z*(Z1+Z2)+Z1*Z2)
15 I2=((E2-E1)*Z+E2*Z1)/(Z*(Z1+Z2)+Z1*Z2)
16
17 phi1=phasemag(I1)  //Phasemag returns the angle of
   complex number in degrees
18 phi2=phasemag(I2)  //Phasemag returns the angle of
   complex number in degrees
19
20 I=I1+I2
21 V=I*Z //Terminal voltage
22 printf('(i) Terminal voltage is %.2f volts at %.1f
```

```
    degrees\n',abs(V),phasemag(V))
23 printf('(ii) Currents are %.2f A at %.0f degrees and
    %.2f A at %.2f degrees\n     Total current is %
    .2f A at %.1f degrees  ',abs(I1),phasemag(I1),abs
    (I2),phasemag(I2),abs(I),phasemag(I))
24
25 P1=abs(V)*abs(I1)*cosd(phi1)
26 P2=abs(V)*abs(I2)*cosd(phi2)
27 printf('\n(iii)Power delivered %.2f watts and  %.2f
    watts',P1,P2)
```

Scilab code Exa 6.34 TO DETERMINE OPEN CIRCUIT VOLTAGE

```
1  clc,clear
2  printf('Example 6.34\n\n')
3
4  X_d=0.8,X_q=0.5  //both per unit
5  R_a=0 //assumed
6  phi=acos(0.8)
7  V_t=1//pu
8  I_a=1 //full-load
9
10 psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
    R_a)            )
11 delta=psi-phi
12 I_d=I_a*sin(psi)
13 I_q=I_a*cos(psi)
14 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
15
16 printf('Open circuit voltage is %.3f p.u.',E_f)
```

Scilab code Exa 6.35 FIND OUTPUT PF AND ARMATURE CURRENT OF SECOND MACHINE OPERATING IN PARALLEL WITH FIRST ALTERNATOR

```
clc,clear
printf('Example 6.35\n\n')

V_L=6600,I_L=110,phi_1=acos(0.9) //lagging
kW=[400,1000,400,300]*10^3
cosphi=[1,0.71,0.8,0.9]
tanphi=tan(acos(cosphi))
kVAR=kW.*tanphi

kW_total=kW(1)+kW(2)+kW(3)+kW(4)
kVAR_total=kVAR(1)+kVAR(2)+kVAR(3)+kVAR(4)

phi_c=atan(kVAR_total/kW_total)  //total power
   factor angle
load_1=sqrt(3)*V_L*I_L*cos(phi_1)

kW_1=load_1 //active component of machine 1
kVAR_1=kW_1*tan(phi_1) //reactive component of
   machine 1
kW_2=kW_total - kW_1 //active component of machine 1
kVAR_2=kVAR_total-kVAR_1 //reactive component of
   machine 2

phi_2=atan(kVAR_2/kW_2)
pf_2=cos(phi_2) //power factor of machine 2

printf('Output of second alternator= %.2f kW',kW_2
   /1000)
printf('\nPower factor of machine 2 = %.4f and
   lagging',pf_2)
```

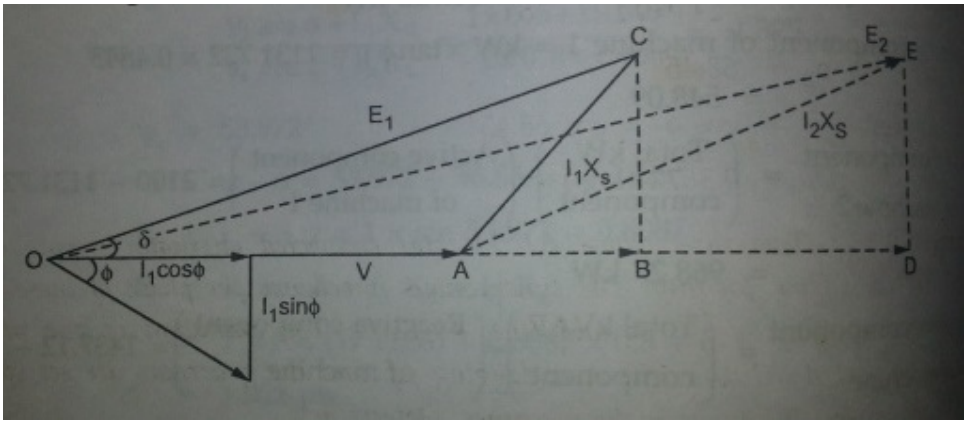


Figure 6.10: TO DETERMINE ALTERED CURRENT AND POWER FACTOR

Scilab code Exa 6.36 TO DETERMINE ALTERED CURRENT AND POWER FACTOR

```
clc,clear
printf('Example 6.36\n\n')

V_L=11000
V_ph=V_L/sqrt(3)
VA=2*10^6,phi=acos(0.8)
I_FL=VA/(sqrt(3)*V_L)
phi_1=acos(0.8)
IX_s=(20/100)*V_ph //product of I and X_s
X_s=IX_s/I_FL
I_1=I_FL
BC=I_1*cos(phi_1)*X_s
AB=I_1*sin(phi_1)*X_s     , OA=V_ph
OC=sqrt( (OA+AB)^2+(BC)^2 ) ,E_1=OC
E_2=1.25*E_1,OE=E_2
DE=BC
```

```
17 AD=sqrt(OE^2-DE^2) -OA              //because OE=sqrt(
     (OA+AD)^2 + (DE)^2   )
18
19 I_2sinphi2=AD/X_s
20 I_2cosphi2=I_1*cos(phi)
21 I_2=sqrt( (I_2cosphi2)^2 + (I_2sinphi2)^2  )
22 phi2=atan( I_2sinphi2/ I_2cosphi2 )
23 new_pf=cos(phi2)
24
25 printf('Machine current is %.2f A \n',I_2)
26 printf('Power factor is  %.4f lagging',new_pf)
```

Scilab code Exa 6.37 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO THREE PHASE STAR CONNECTED ALTERNATORS OPERATING IN PARALLEL

```
1  clc,clear
2  printf('Example 6.37\n\n')
3  //note that a new function p2z has been defined
     below for direct representation of complex
     numbers in polar form
4  function [FUN] = p2z(RRRR,Theeeta)
5   FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
6  endfunction
7
8  P_out=3000*10^3
9  V_L=6.6*10^3,V_ph=V_L/sqrt(3)
10 phi=acos(0.8)
11 I_L=p2z(P_out/(sqrt(3)*V_L*cos(phi)),-1*(180/%pi)*
     phi)
12
13 P_out1=P_out/2
14 I_L1=150  //given
15 phi_L1=acos(    P_out1/(sqrt(3)*V_L*I_L1)  )
16 I_L1=p2z(I_L1,-1*(180/%pi)*phi_L1)
```

```
17
18 I_L2=I_L-I_L1
19 pf_2=cosd(phasemag(I_L2))
20 Z_1=complex(0.5,10)
21 I_1=I_L1
22 E_1=V_ph + I_1*Z_1
23 delta_1=(%pi/180)*phasemag(E_1)     //load angle of
     alternator 1
24 E_1L=sqrt(3)*E_1
25
26 Z_2=complex(0.4,12)
27 I_2=I_L2
28 E_2=V_ph + I_2*Z_2
29 delta_2=(%pi/180)*phasemag(E_2)     //load angle of
     alternator 2
30
31 printf('Part(i)\nCurrents are %.0f A at %.1f degrees
     and %.1f A at %.1f degrees\nTotal current is %.0
    f at %.2f\n',abs(I_L1),phasemag(I_L1),abs(I_L2),
    phasemag(I_L2),abs(I_L),phasemag(I_L))
32 printf('Part(ii)\nPower factor is %.4f and lagging\n
    ',cos(phi_L1))
33 printf('Part(iii)\nemf are %.2f V at %.2f degrees
    and %.4f V at %.0f degrees\n',abs(E_1),phasemag(
    E_1),abs(E_2),phasemag(E_2))
34 printf('Part(iv)\nPower angles are %.2f degrees and
    %.0f degrees \n',(180/%pi)*delta_1,(180/%pi)*
    delta_2)
```

Scilab code Exa 6.38 TO DETERMINE THE kW OUTPUT AND POWER FACTOR OF EACH OF THE SYNCHRONOUS GENERATOR

```
1 clc,clear
```

The phase diagram for alternator 1 is

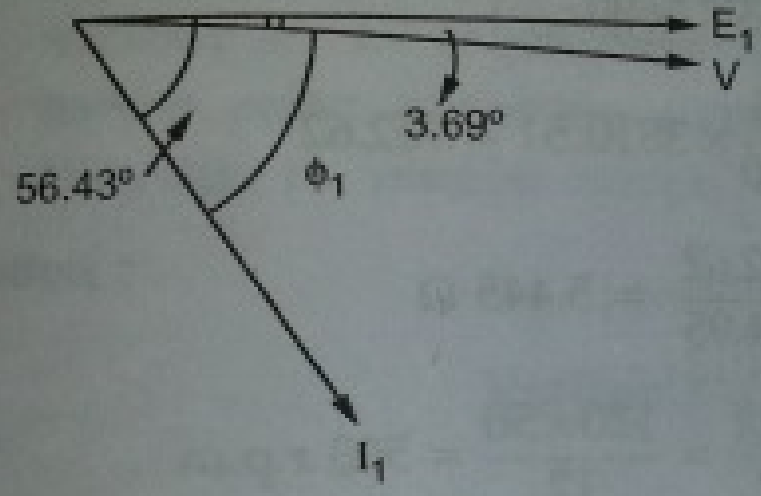


Fig. 6.66

The phasor diagram for alternator 2 is

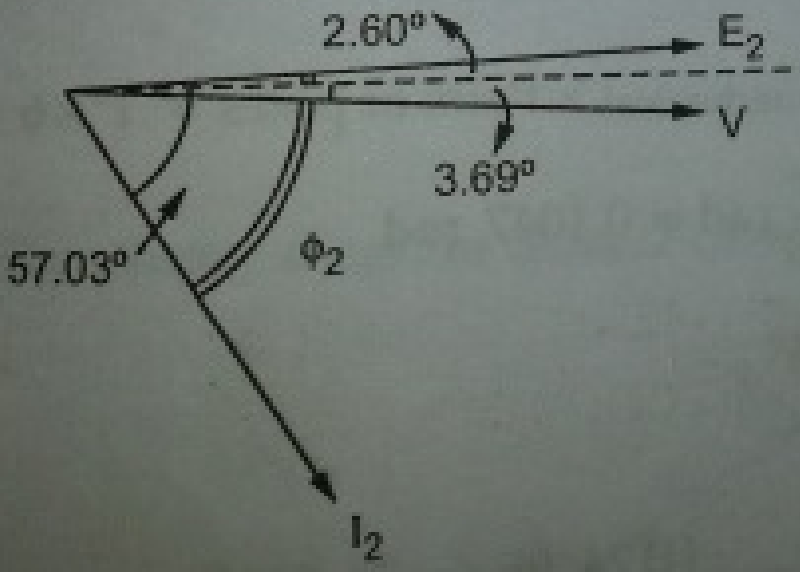


Fig. 6.67

Figure 6.11: TO DETERMINE THE kW OUTPUT AND POWER FACTOR OF EACH OF THE SYNCHRONOUS GENERATOR

```
2 printf('Example 6.38\n\n')
3
4 Z1=complex(0.2,2)
5 Z2=Z1
6 ZL=complex(3,4)
7 Z=ZL
8 E1=complex(2000,0)
9 E2=complex(2200,100)
10
11 I1=((E1-E2)*Z+E1*Z2)/(Z*(Z1+Z2)+Z1*Z2)
12 I2=((E2-E1)*Z+E2*Z1)/(Z*(Z1+Z2)+Z1*Z2)
13
14 IL=I1+I2
15 V=IL*Z //Terminal voltage
16
17 phi1=phasemag(V)-phasemag(I1)  //Phasemag returns
     the angle of complex number in degrees
18 phi2=phasemag(V)-phasemag(I2)  //Phasemag returns
     the angle of complex number in degrees
19
20 Pout1=sqrt(3)*sqrt(3)*abs(V)*abs(I1)*cosd(phi1)
21 Pout2=sqrt(3)*sqrt(3)*abs(V)*abs(I2)*cosd(phi2)
22 printf('\nPower delivered is %.2f kW and %.2f kW at
     power-factors %.4f lag and %.4f lag respectively'
     ,Pout1/1000,Pout2/1000,cosd(phi1),cosd(phi2))
```

Scilab code Exa 6.39 TO CALCULATE SYNCHRONISING POWER PER MECHANICAL DEGREE OF DISPLACEMENT

```
1 clc,clear
2 printf('Example 6.39\n\n')
3
4 f=50
5 P=12
6 V_L=6600
```

```
7  V_ph=V_L/sqrt(3)
8  VA=2000*10^3
9  I_FL=VA/(sqrt(3)*V_L)
10
11 IX_s=(25/100)*V_ph //product of I and X_s
12 X_s=complex(0,IX_s/I_FL)
13 N_s=12*f/P //in rpm
14 delta_dash_mech=(%pi/180) //phase displacemnt in
      degree mechanical
15 delta_dash_elec=delta_dash_mech*(P/2) //phase
      displacemnt in degree electrical
16
17 phi=acos(0.8) //lag
18 I=complex(I_FL*cos(-1*phi),I_FL*sin(-1*phi))
19 V=V_ph
20 E=V + I*X_s
21 delta=phasemag(E)*(%pi/180)
22 P_SY=abs(E)*abs(V)*cos(delta)*sin(delta_dash_elec)/
      abs(X_s)
23 P_SY_total=3*P_SY
24 printf('\nSynchronising power is %.2f kW',P_SY/1000)
25 printf('\nTotal synchronising power is %.2f kW',
      P_SY_total/1000)
```

Scilab code Exa 6.40 TO DETERMINE THE ALTERNATOR CURRENT AND POWER FACTOR

```
1  clc,clear
2  printf('Example 6.40\n\n')
3
4  V_L=22000
5  V_ph=V_L/sqrt(3)
6  power=230*10^6
```

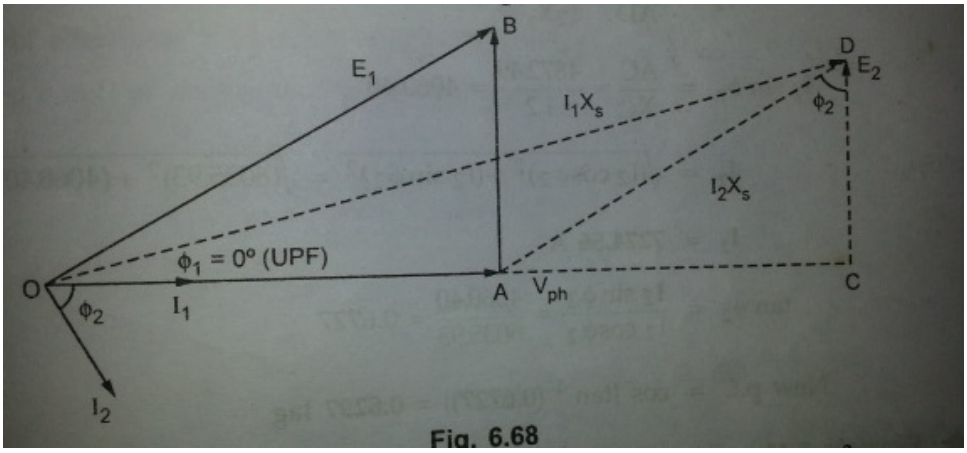


Figure 6.12: TO DETERMINE THE ALTERNATOR CURRENT AND POWER FACTOR

```
7  phi=acos(1)
8  I_FL=power/(sqrt(3)*V_L*cos(phi))
9  I_1=I_FL
10 X_s=1.2
11
12 E_1=sqrt( V_ph^2 + (I_1*X_s)^2 )
13 E_2=1.3*E_1
14 AC=  sqrt( E_2^2-(I_1*X_s)^2  )   -V_ph      //
     because E^2=(V_ph+AC)^2+(I_1*X_s)^2
15 I2X_S=AC
16
17 I_2cosphi2=I_1  //because phi_2=acos(I_1/I_2) //from
     ACD
18 I_2sinphi2=AC/X_s
19 I_2=sqrt( (I_2cosphi2)^2 + (I_2sinphi2)^2  )
20 phi2=atan( I_2sinphi2/ I_2cosphi2 )
21 new_pf=cos(phi2)
22
23 printf('Machine current is %.2f A \n',I_2)
24 printf('Power factor is  %.4f and lagging',new_pf)
```

Scilab code Exa 6.41 TO DETERMINE CERTAIN CHARACTERISTICS RELATED TO EACH OF THE 2 ALTERNATORS

```
clc,clear
printf('Example 6.41\n\n')
//note that a new function p2z has been defined
   below for direct representation of complex
   numbers in polar form
function [FUN] = p2z(RRRR,Theeeta)
 FUN = RRRR.*exp(%i*%pi*Theeeta/180.);
endfunction

P_out=1500*10^3
V_L=3.3*10^3
phi=acos(0.8)
I_L=p2z(P_out/(sqrt(3)*V_L*cos(phi)),-1*acosd(0.8))

I_L1_magnitude=150  //given
P_out1=(3*10^6)/2 //because load is EQUALLY shared
   between 2 alternators
pf_L1=P_out1/(sqrt(3)*2*V_L*I_L1_magnitude)  //
   operating pf of alternator 1
phi1=acosd(pf_L1)
I_L1=p2z(I_L1_magnitude,-1*phi1)
I_L2=I_L-I_L1   //because I_L=I_L1 + I_L2
pf_L2=cosd(phasemag(I_L2))

V_ph=6.6*10^3/sqrt(3)
Z_1=complex(0.5,10)
I_1=I_L1
E_1= V_ph + I_1*Z_1
delta_1=phasemag(E_1)  //load angle of alternator 1
I_2=I_L2

```

```
28 Z_2=complex(0.4,12)
29 E_2= V_ph + I_2*Z_2
30 delta_2=phasemag(E_2)  //load angle of alternator 1
31
32 printf('for machine 1\ncurrent is %.0f A at %.2f
      degrees\nPower factor of %.4f lag\ninduced emf of
       %.2f V\nload angle of %.2f degrees',abs(I_L1),
      phasemag(I_L1),pf_L1,abs(E_1),delta_1)
33 printf('\n\nfor machine 2\ncurrent is %.1f A at %.1f
       degrees\nPower factor of %.4f lag\ninduced emf
      of %.2f V\nload angle of %.0f degrees',abs(I_L2),
      phasemag(I_L2),pf_L2,abs(E_2),delta_2)
```

Scilab code Exa 6.42 TO CALCULATE THE EXCITATION VOLTAGE

```
1  clc,clear
2  printf('Example 6.42\n\n')
3
4  V_l=230
5  VA=5*10^3
6  X_d=12,X_q=7,R_a=0 //armature resistance and
      synchronous reactance of direct,quadrature axis
7  phi=acos(1)
8
9  I_l=VA/(V_l*sqrt(3))
10 V_ph=V_l/sqrt(3)
11 V_t=V_ph,I_a=I_l
12
13 psi=atan(  (V_t*sin(phi)+I_a*X_q)/(V_t*cos(phi)+I_a*
      R_a)            )
14 delta=psi-phi
15 I_d=I_a*sin(psi)
16 I_q=I_a*cos(psi)
17 E_f=V_t*cos(delta)+I_d*X_d+I_q*R_a
18
```

```
19 printf('Excitation voltage is %.3f V ',E_f)
```

Scilab code Exa 6.43 TO DETERMINE EXCITATION EMF AT CERTAIN POWER FACTOR AND MAXIMUM LOAD THE MOTOR CAN SUPPLY AT NO EXCITATION

```
1 clc,clear
2 printf('Example 6.43\n\n')
3
4 V_l=6.6*10^3
5 V_t=V_l/sqrt(3)
6 X_d=23.2,X_q=14.5,R_a=0 //armature resistance and
    synchronous reactance of direct,quadrature axis
7 VA=1800*10^3
8 phi=acos(0.8) //lag
9
10 I_a=VA/(V_l*sqrt(3))
11
12 psi=atan(  (V_t*sin(phi)-I_a*X_q)/(V_t*cos(phi)-I_a*
    R_a)  ) //minus sign in numerator and denomenator
     for motors
13 delta=psi+phi
14 I_d=I_a*sin(psi)
15 I_q=I_a*cos(psi)
16 E_f=V_t*cos(delta)-I_d*X_d-I_q*R_a
17 printf('Excitation emf = %.4f V\n',E_f)
18 //P_m= ( V_t*E_f*sin(delta)/X_d  ) + ((1/X_q)-(1/X_d
    ))*0.5*sin(2*delta)*V_t^2
19 //P_m=0.4996*cos(delta)+0.1877*sin(2*delta)
20 //for maximum power output, differenciate and equate
     to zero
21
22 delta_max=63.4 //degree
23
24 P_m_max=((1/X_q)-(1/X_d))*0.5*sind(2*delta_max)*V_t
```

```
    ^2    //Maximuum load supplied with E_f=0
25 printf('Maximum load the motor can supply is %.4f MW
    per phase ',P_m_max*10^-6       )
```

Chapter 7

Synchronous Motors

Scilab code Exa 7.1 TO CALCULATE THE BACK EMF INDUCED IN THE MOTOR FOR VARIOUS POWER FACTORS

```
1  clc,clear
2  printf('Example 7.1\n\n')
3
4  V_l=400
5  R_a=0.2,X_s=2  //armature resistance and synchronous
       reactance
6  I_L=25
7  I_aph=I_L
8  V_ph=V_l/sqrt(3)
9  Z_s=complex(R_a,X_s) //synchronous impedance
10 theta=(%pi/180)*phasemag(Z_s) //Phasemag returns the
       angle in degrees not radians
11 E_Rph=I_aph*abs(Z_s)
12
13 //case 1
14 phi=acos(0.8) //lagging
15 E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*E_Rph*V_ph*cos
     (theta-phi)  )
16 printf('\n(i)Back EMF induced with 0.8 lagging pf is
       %.3f V\n',E_bph)
```

```
17
18 //case 2
19 phi=acos(0.9) //leading
20 E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*E_Rph*V_ph*cos
     (theta+phi)  )
21 printf('(ii)Back EMF induced with 0.8 lagging pf is
     %.3f V\n',E_bph)
22
23 //case 3
24 phi=acos(1)
25 E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*E_Rph*V_ph*cos
     (theta)  )
26 printf('(iii)Back EMF induced with 0.8 lagging pf is
     %.3f V',E_bph)
```

Scilab code Exa 7.2 TO DETERMINE THE OPERATING POWER FACTOR FOR DIFFERENT GENERATED EMF

```
1 clc,clear
2 printf('Example 7.2\n\n')
3
4 V_l=500
5 R_a=0.4,X_s=4 //armature resistance and synchronous
     reactance
6 Z_s=complex(R_a,X_s)//synchronous impedance
7 theta=(%pi/180)*phasemag(Z_s)//phasemag returns
     angle in degrees,not radians
8 V_ph=V_l/sqrt(3)
9 I_l=50
10 I_aph=I_l
11 E_Rph=I_aph*abs(Z_s)
12
13 //case 1
14 E_bline=600
15 E_bph=E_bline/sqrt(3)
```

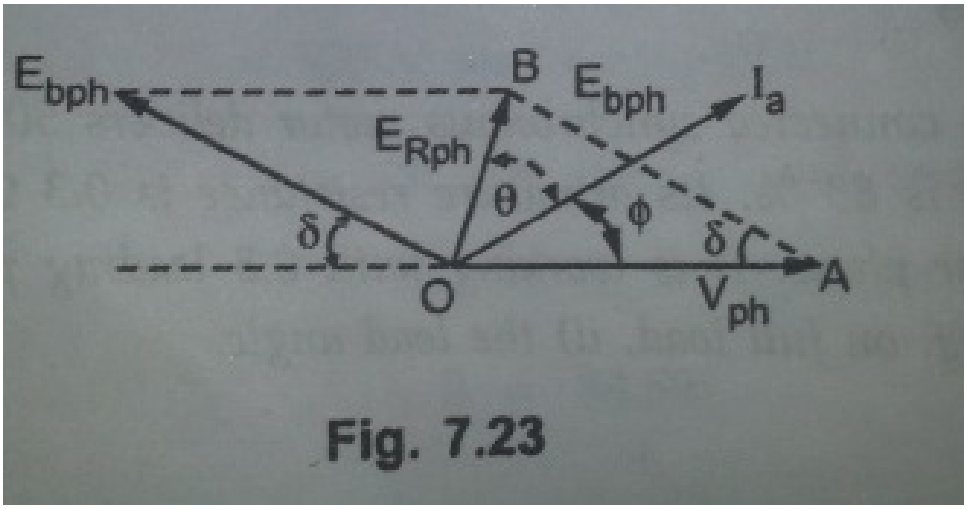


Figure 7.1: TO DETERMINE GENERATED EMF ON FULL LOAD AND THE LOAD ANGLE

```
16 phi=acos( (-E_bph^2 + E_Rph^2 + V_ph^2 )/(2*E_Rph*
      V_ph) ) -theta     //leading
17 //because E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*
      E_Rph*V_ph*cos(theta+phi)  )
18 printf('(i)power factor is %.4f leading\n',cos(phi))
19
20 //case 2
21 E_bline=380
22 E_bph=E_bline/sqrt(3)
23 phi= theta-acos( (-E_bph^2 + E_Rph^2 + V_ph^2 )/(2*
      E_Rph*V_ph) )      //leading
24 //because E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*
      E_Rph*V_ph*cos(theta-phi)
25 printf('(ii)power factor is %.4f lagging\n',cos(phi)
      )
```

Scilab code Exa 7.3 TO DETERMINE GENERATED EMF ON FULL LOAD AND THE LOAD ANGLE

```
clc,clear
printf('Example 7.3\n\n')

V_L=6600
P_out=500*10^3
eta=83/100 //efficiency
R_a=0.3,X_s=3.2 //armature resistance and
   synchronous reactance
Z_s=complex(R_a,X_s)  //synchronous impedance
theta=(%pi/180)*phasemag(Z_s)  //phasemag returns
   the angle in degrees not radians
phi=acos(0.8) //leading
V_ph=V_L/sqrt(3)
P_in=P_out/eta

I_L=   P_in/ (sqrt(3) * V_L *  cos(phi) )
// because P_in=sqrt(3) * V_L * I_L * cos(phi)
I_aph=I_L
E_Rph=I_aph*abs(Z_s)
E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*E_Rph*V_ph*cos
   (theta+phi)  )
printf('(i) Generated EmF on full loaad is %.2f V\n'
   ,E_bph)

delta= asind(  (E_Rph/E_bph)*sin(theta+phi)  )
//This is obtained after applying sune rule to
   triangle OAB from thre phasor diagram
printf('(ii) load angle is %.2f degrees',delta)
```

Scilab code Exa 7.4 TO DETERMINE CURRENT DRAWN BY THE MOTOR AND ITS FULL LOAD EFFICIENCY

```
clc,clear
printf('Example 7.4\n\n')

V_L=500,V_ph=V_L/sqrt(3)
phi=acos(0.9)  //lagging
output_power=17*10^3
R_a=0.8 //armaature reactance
mechanical_losses=1300 //mechanical losses is W
P_m=output_power+mechanical_losses   //gross
   mechanical power developed

//  P_m= input_power - stator losses
//  input_power= 3* V_ph * I_aph * cos(phi)
//  Stator losses= 3*I_aph^2*R_a
// solving above equations we get 2.4 I_a^2 -
   779/.4225*I_a + 18300 = 0
I_a_eqn=[2.4 -779.4225 18300]
I_a_roots=roots(I_a_eqn)
I_a=I_a_roots(2)  //neglecting higher value
I_aph=I_a
printf('Current drawn by the motor is %.3f A\n',I_a)

input_power= 3* V_ph * I_aph * cos(phi)
eta=100*output_power/input_power
printf('Full load efficiency is %.2f percent',eta)
```

Scilab code Exa 7.5 TO DETERMINE kVA RATING OFDESIRED SYNCHRONOUS MOTOR AND ITS OPERATING POWER FACTOR

```
clc,clear
printf('Example 7.5\n\n')

//subscript 1 is for industrial load and 2 for
```

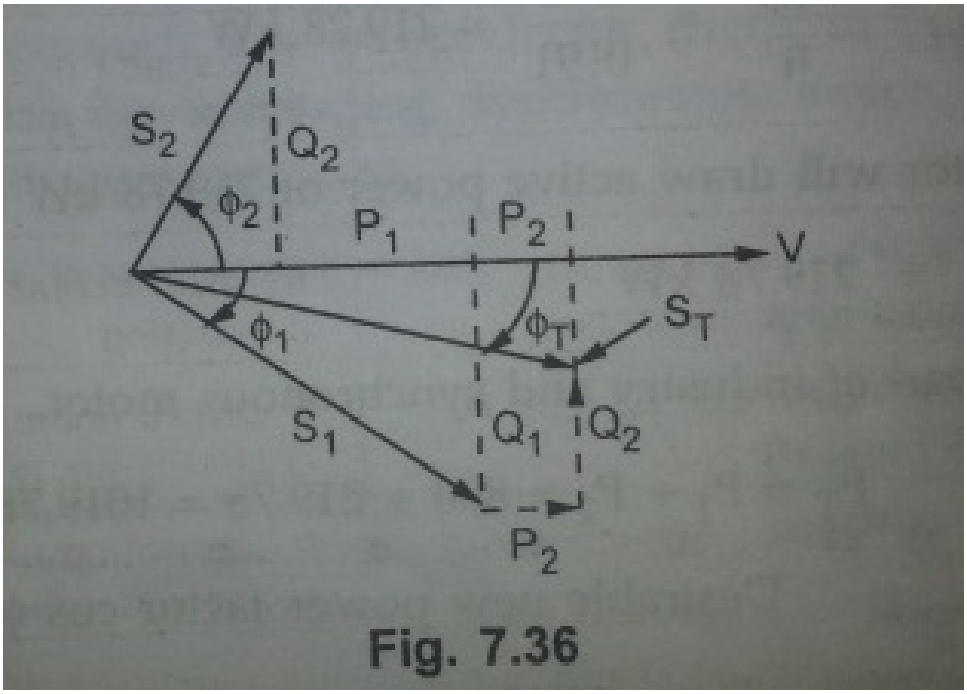


Figure 7.2: TO DETERMINE kVA RATING OFDESIRED SYNCHRONOUS MOTOR AND ITS OPERATING POWER FACTOR

```
    synchronous motor
P_1=800 // Active power in KW
phi_1=acos(0.6) //lagging
Q_1=P_1*tan(phi_1)   //reactive power by load 1

output_power=200
eta=91/100 //efficiency of synchronous motor
input_power= output_power/eta
P_2=input_power// active power drawn by synchronous
    motor
P_T=P_1 + P_2  //combined total load of industry and
     synchronous motor
phi_T=acos(0.92 )//lagging
Q_T=P_T* tan(phi_T)  //from power triangle
Q_2= Q_T - Q_1  //it turns out to be negative
    indicating its leading nature
S_2=sqrt( P_2^2 + Q_2^2    )
printf('Desired kVA rating of Synchronous motor  is
    %.3f kVA',S_2)

phi_2= atan (Q_2/P_2)
printf('\nPower factor of synchronous motor is %.4f
    LEADING',cos(phi_2))
```

Scilab code Exa 7.6 TO DETERMINE INDUCED EMF ON FULL LOAD

```
clc,clear
printf('Example 7.6\n\n')

V_L=400
output_power=37.3*1000 //Watts on full load
Z_s=complex(0.2,1.6) //synchronous impedance
theta=(%pi/180)*phasemag(Z_s) //phase mag returns
    the angle in degrees and not raidians
phi=acos(0.9) //leading
```

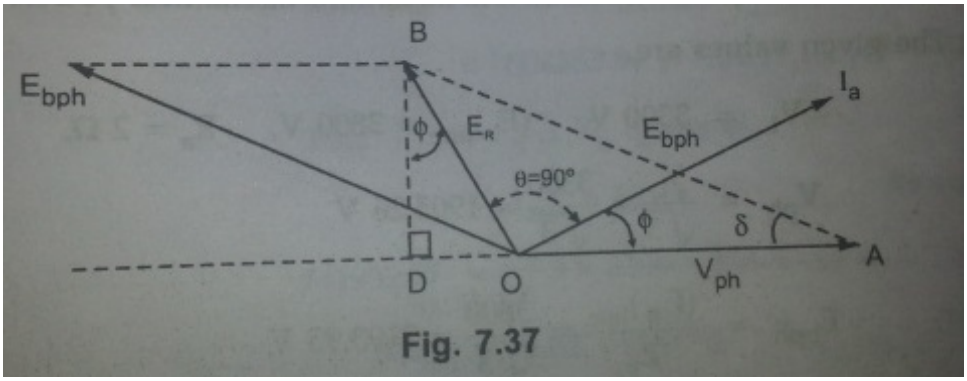


Figure 7.3: TO CALCULATE MOTOR POWER FACTOR AND CURRENT DRAWN BY IT

```
9  V_ph=V_L/sqrt(3)
10 eta=88 //efficiency in percentage
11 input_power=100*output_power/eta
12 I_L=input_power/(sqrt(3)*V_L*cos(phi))
13 I_aph=I_L
14 E_Rph=I_aph*abs(Z_s)
15
16 E_bph= sqrt( (E_Rph)^2 + (V_ph)^2  -2*E_Rph*V_ph*cos
     (theta+phi)  )
17 E_line=sqrt(3)*E_bph
18
19 printf('Induced EMF is %.2f V and its line value is
     %.2f V',E_bph,E_line)
```

Scilab code Exa 7.7 TO CALCULATE MOTOR POWER FACTOR AND CURRENT DRAWN BY IT

```
1 clc,clear
2 printf('Example 7.7\n\n')
3
```

```
4 V_L=400
5 input_power=20*1000
6 R_a=0,X_s=4 //armature reactance and synchronous
    reactance
7 Z_s=complex(R_a,X_s)   //synchronous impedance
8 theta=(%pi/180)*phasemag(Z_s)  //phase mag returns
    the angle in degrees and not raidians
9 V_ph=V_L/sqrt(3)
10 E_bline=550 //star connection
11 E_bph=E_bline/sqrt(3)
12
13 I_a_cos_phi=input_power/(sqrt(3)*V_L)  //product of
    I_a and cos(phi)
14 I_a_sin_phi= (   sqrt(E_bph^2- (abs(Z_s)*I_a_cos_phi
    )^2   )  -V_ph )/abs(Z_s)//from triangle DAB
15 phi=atan(I_a_sin_phi/I_a_cos_phi)
16 I_a=I_a_cos_phi/cos(phi)
17
18 printf('Motor power fctor is %.3f Leading\n',cos(phi
    ))
19 printf('Current drawn by the motor is %.2f A',I_a)
```

Scilab code Exa 7.8 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER

```
1 clc,clear
2 printf('Example 7.8\n\n')
3 printf('Answer in part(1) mismatched because of
    improper approximation in book\n\n')
4
5 V_L=3300, V_ph=V_L/sqrt(3)
6 R_a=2,X_s=18 //armature reactance and synchronous
    reactance
```

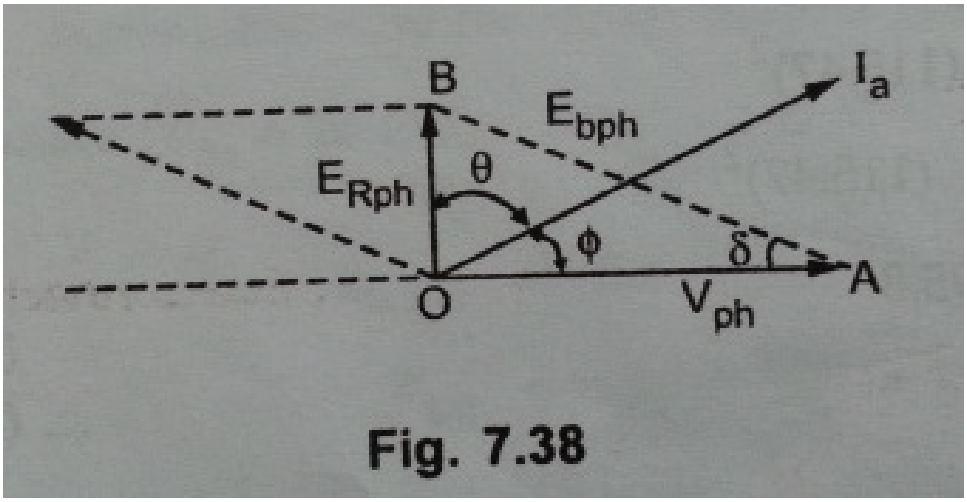


Figure 7.4: TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER

```
7  Z_s=complex(R_a,X_s)//synchronous impedance
8  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
       angle in degrees not radians
9  E_bline=3800,E_bph=E_bline/sqrt(3)
10
11 //part(i)
12 P_m_max = (E_bph*V_ph/abs(Z_s))- (E_bph^2/abs(Z_s))*
       cos(theta)
13 printf('(i)Max total mechanical power developed that
        motor can develop is %.2f W per phase\n',P_m_max
       )
14
15 //part(ii)
16 //from phasor diagram, applying cosine rule to
       triangle OAB
17 E_Rph=sqrt(  E_bph^2 + V_ph^2 -2*E_bph*V_ph*cos(
       theta) )
18 I_aph= E_Rph/abs(Z_s)
19 printf('(ii)Current at max power developed is %.1f A
       \n',I_aph)
20
```

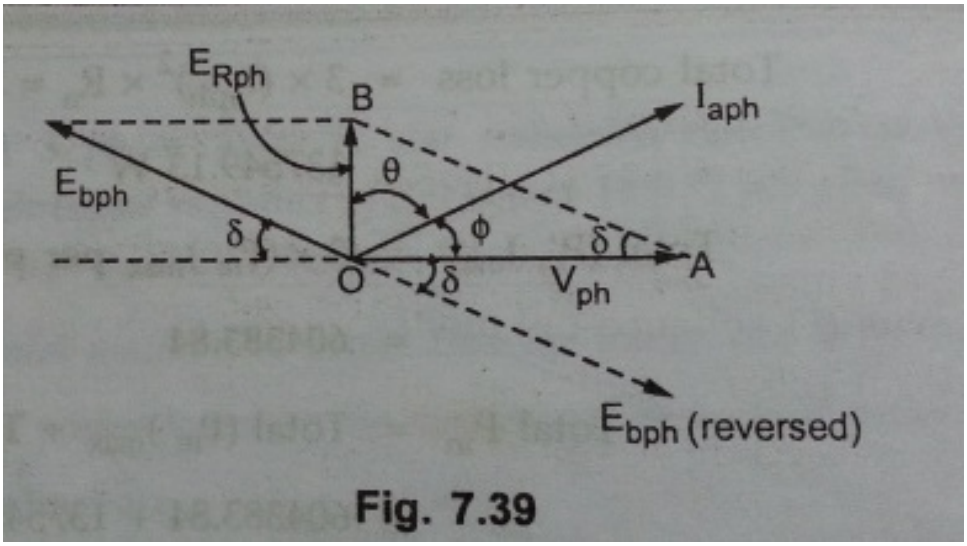


Figure 7.5: TO DETERMINE EMF AND MECHANICAL POWER DEVELOPED

```
21 copper_loss=3* I_aph^2 * R_a
22 P_in_max_total=3 * P_m_max //input power at max
      power developed
23 total_P_in= P_in_max_total + copper_loss //total
      input power
24 pf=total_P_in/(sqrt(3)*I_aph*V_L)
25 printf('Power factor at max power developed is %.3f
      leading',pf)
```

Scilab code Exa 7.9 TO DETERMINE EMF AND MECHANICAL POWER DEVELOPED

```
1 clc,clear
2 printf('Example 7.9\n\n')
3
```

```
4 V_L=500
5 R_a=0.03,X_s=0.3 //armature reactance and
    synchronous reactance
6 Z_s=complex(R_a,X_s)//synchronous impedance
7 theta=(%pi/180)*phasemag(Z_s) //phasemag returns
    angle in degrees,not radians
8 phi=acos(0.8)
9 eta=93/100
10 output_power=100*746
11 input_power=output_power/eta
12 I_L=input_power/(sqrt(3)*V_L*cos(phi))
13 I_aph=I_L
14 E_Rph=I_aph*abs(Z_s)
15 //from the phasor diagram
16 E_bph =   sqrt( E_Rph^2 + (V_L/sqrt(3))^2 - 2*E_Rph
    *(V_L/sqrt(3))*cos(phi+theta)  )
17
18 cu_losses=3*(I_aph)^2*R_a //total copper losses
19 P_m= input_power - cu_losses  //total mechanical
    power developed
20
21 printf('EMF developed per phase is %.4f V \nTotal
    mechanical power developed is %.1f watts',E_bph,
    P_m)
```

Scilab code Exa 7.10 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE MESH CONNECTED SYNCHRONOUS MOTOR

```
1 clc,clear
2 printf('Example 7.10\n')
3 printf('Answer might mismatch because of improper
    approximation done in book\n\n')
4
5 V_L=415
6 V_ph=V_L //due to delta connection
```

```
E_bline=520
R_a=0.5,X_s=4 //armature reactance and synchronous
   reactance
Z_s=complex(R_a,X_s) //synchronous impedance
theta=(%pi/180)*phasemag(Z_s) //phasemag returns
   angle in degrees,not radians

delta=theta //for maximum power
P_m_max = (E_bline*V_ph/abs(Z_s))- (E_bline^2/abs(
   Z_s))*cos(theta)
P_m_max_total= 3* P_m_max
fi_loss=1000  //frictional and iron losses
P_out_total = P_m_max_total-fi_loss

HP_output= P_out_total/746 //converting watts to
   horse power
printf('HP output for maximum power output is %.2f
   HP\n',HP_output)

//from the phasor diagram
E_Rph=sqrt(  E_bline^2 + V_ph^2 -2*E_bline*V_ph*cos(
   delta) )
I_aph= E_Rph/abs(Z_s)
I_L=I_aph*sqrt(3)
printf('Line current is %f A\n',I_L)
cu_loss_total=3*(I_aph)^2*R_a  //total copper losses
input_power=P_m_max_total+ cu_loss_total
pf=input_power/(sqrt(3)*I_L*V_L)  //leading
printf('Power factor for maximum power output is %.2
   f leading \n',pf)

eta=100*P_out_total /input_power
printf('Efficiency for maximum power output is %.2f
   percent',eta)
```

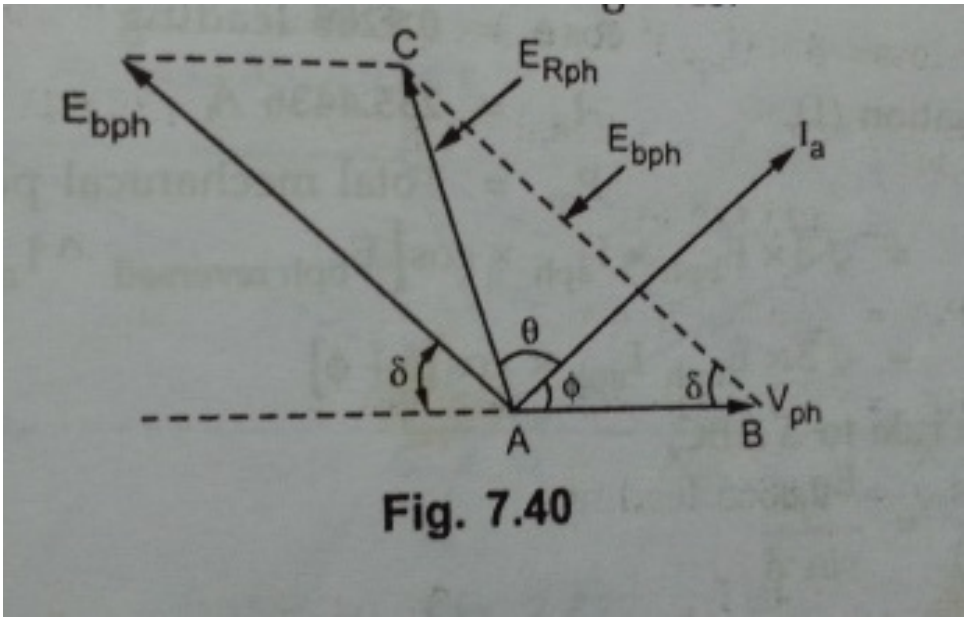


Figure 7.6: TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

Scilab code Exa 7.11 TO DETERMINE CERTAIN QUANTITIES RELATED TO 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
1  clc,clear
2  printf('Example 7.11\n\n')
3
4  P=8, f=50 //Pole and frequency
5  N_s=120*f/P //synchronous speed
6  V_L=6.6*10^3 , V_ph=V_L/sqrt(3)
7  Z_s=complex(0.66,6.6) //synchronous impedance
8  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
      angle in degree , not radians
9  E_bph=4500
10 input_power=2500*10^3
```

```
11 I_a_cosphi=input_power/(sqrt(3)*V_L)   //Its product
       of I_a and cos(phi);I_a=I_l for star conneted
      load
12
13 //applying cosine rule to triangle ABC from phasor
      diagram and solve
14 //tan(phi)^2 + 5.2252 tan(phi)-2.2432=0
15 p=[1 5.2252 -2.2432]
16 tan_phi=roots(p)
17 phi=atan(tan_phi(2))
18 pf=cos(phi)
19 I_a= I_a_cosphi/ cos(phi)
20
21 //apply sine rule to triangle ABC
22 delta=   asin(I_a*abs(Z_s)*sin(theta+phi)/E_bph)
23 P_m=3*E_bph*I_a*cos(delta+phi)
24 T_g = P_m/(2*%pi*N_s/60)
25 printf('(i)Torque developed is %f N-m\n',T_g)
26 printf('(ii)Input current is %.4f A\n',I_a)
27 printf('(iii)Power factor is %.4f leading\n',pf)
28 printf('(iv)Power angle is %.2f degrees ',(180/%pi)*
      delta)
```

Scilab code Exa 7.12 TO DETERMINE LOAD ANGLE ARMATURE CURRENT AND PF WHEN EXCITATION IS CHANGED

```
1 clc,clear
2 printf('Example 7.12\n\n')
3
4 input_power=15*10^3
5 V_L=400,V_ph=V_L/sqrt(3)
6 E_b=480,E_bph=E_b/sqrt(3)
7 Z_s=complex(1,5) //synchronous impedance
```

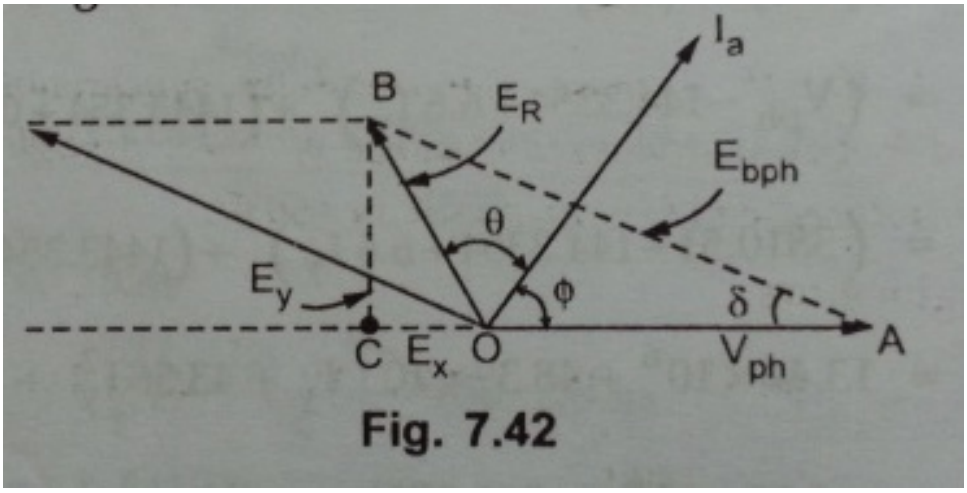


Figure 7.7: TO DETERMINE LOAD ANGLE ARMATURE CURRENT AND PF WHEN EXCITATION IS CHANGED

```
8  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
       angle in degree , not radians
9
10 I_a_cosphi=input_power/(sqrt(3)*V_L)  //product of
       I_a & cos(phi)
11 //Applying cosine rule to triangle OAB and solving
12 //tan(phi)^2+ 4.101*tan(phi)-1.7499=0
13 p=[1,4.101,-1.7449]
14 tan_phi=roots(p)
15 phi=atan(tan_phi(2))  //ignoring negative vaule
16 I_a= I_a_cosphi/ cos(phi)
17
18 //applying sine rule to Triangle OAB
19 delta=asin( I_a*abs(Z_s)* sin(theta+phi)/E_bph )
20 printf('Load angle is %.1f degrees',delta*(180/%pi))
21 printf('\nArmature current is %.4f A',I_a)
22 printf('\nPower factor is %.3f leading',cos(phi))
```

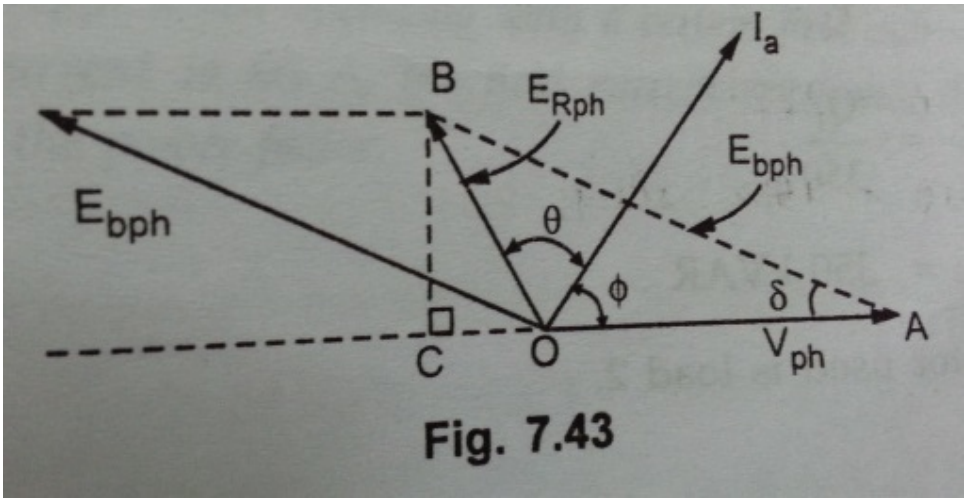


Figure 7.8: TO CALCULATE CURRENT AND PF IF INDUCED EMF IN SYNCHRONOUS MOTOR GETS INCREASED

Scilab code Exa 7.13 TO CALCULATE CURRENT AND PF IF INDUCED EMF IN SYNCHRONOUS MOTOR GETS INCREASED

```
1  clc,clear
2  printf('Example 7.13\n\n')
3
4  V_L=400,V_ph=V_L/sqrt(3)
5  E_b=460,E_bph=E_b/sqrt(3)
6  input_power=3.75*10^3
7  Z_s=complex(1,8) //synchronous impedance
8  theta=(%pi/180)*phasemag(Z_s)  //phasemag returns
       angle in degree , not radians
9  I_L_cos_phi = input_power/(sqrt(3)*V_L)
10
11 //Applying cosine rule to triangle OAB and solving
       further
12 //tan(phi)^2 + 458.366*tan(phi) -450.65 =0
```

```
13 p=[1,458.366,-450.65]
14 tan_phi=roots(p)
15 phi=atan(tan_phi(2))  //ignoring negative value
16 printf('Required power factor is %.4f leading',cos(
      phi))
17 I_L=I_L_cos_phi /cos(phi)
18 printf('\nRequired current is %.4f A',I_L)
```

Scilab code Exa 7.14 TO FIND kVA RATING OF SYNCORONOUS MOTOR

```
1  clc,clear
2  printf('Example 7.14\n\n')
3
4  //subscript 1 indicates induction motor 1
5  P_1=350
6  phi_1=acos(0.7071) //lagging
7  Q_1=P_1*tan(phi_1)//from power triangle
8
9  //subscript 2 indicates induction motor 2
10 P_2=190
11
12 //subscript T indicates total
13 P_T=P_1+P_2
14 phi_T=acos(0.9) //lagging
15 Q_T=P_T*tan(phi_T)
16
17 Q_2=Q_T-Q_1
18 kva_rating=sqrt(P_2^2+ Q_2^2)
19 printf('kVA rating of synchronous motor is %.2f kVA'
      ,kva_rating)
```

Scilab code Exa 7.15 TO FIND GROSS TORQUE DEVELOPED AND PF WITH CHANGING CURRENT AND LOAD TORQUE

```
clc,clear
printf('Example 7.15\n\n')

V_L=400, V_ph=V_L/sqrt(3)
Pole=6,f=50
R_a=0.2,X_s=3 //armature reactance and synchronous
   reactance
Z_s=complex(R_a,X_s)//synchronous impedance
theta=phasemag(Z_s)*(%pi/180)  //phasemag retuens
   angle in degrees.not radians
N_s=120*f/Pole  //synchronous speed

//subscript 1' refers to load 1
I_a1=20
phi_1=acos(1)
E_R1= I_a1* abs(Z_s)
E_bph =  sqrt( E_R1^2 + V_ph^2 - 2*E_R1*V_ph*cos(
   phi_1+theta)  )

//subscript 2' refers to load 2
I_a2=60
E_R2= I_a2* abs(Z_s)
phi_2= acos ((E_R2^2 + V_ph^2 -E_bph^2 )/(2*E_R2*
   V_ph))   -theta   //new power factor

input_power=sqrt(3)*V_L*I_a2*cos(phi_2)
cu_loss=3*I_a2^2*R_a
P_m=input_power-cu_loss
T_g =P_m /(2*%pi*N_s/60)   //gross mechanical power
   developed

printf('Gross torque developed is %.4f N-m and new
   power factor is %.4f lagging',T_g,cos(phi_2))
```

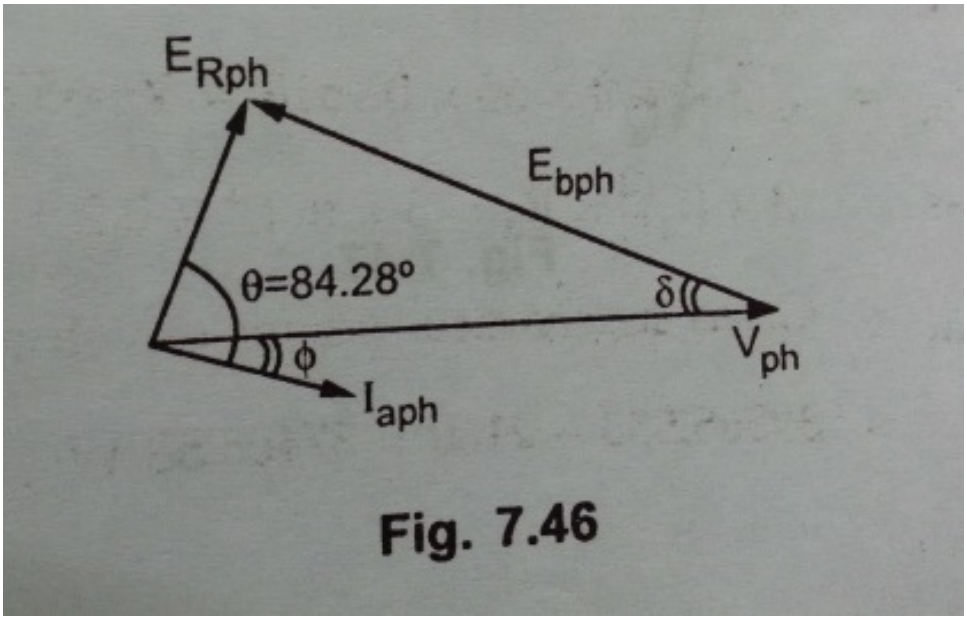


Figure 7.9: TO DETERMINE ARMATURE CURRENT AND PF OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

Scilab code Exa 7.16 TO DETERMINE ARMATURE CURRENT AND PF OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
1 clc,clear
2 printf('Example 7.16\n\n')
3
4 V_L=3300
5 V_ph=V_L/sqrt(3)
6 E_bph=V_ph
7 Z_s=complex(0.5,5) //synchronous impedance
8 theta=(%pi/180)*phasemag(Z_s) //phasemag returns
    angle in degrees, not radians
```

```
P=8,f=50 //pole and frequency
delta_mech=3 //mechanical angle in degrees by which
   rotor is behind
delta_elec=(P/2)*delta_mech  //delta mech converted
   to electrical degrees
E_Rph=sqrt(  E_bph^2 + V_ph^2 -2*E_bph*V_ph*cosd(
   delta_elec) )
I_aph= E_Rph/abs(Z_s)

//from the phasor diagram
phi=theta- asin( sind(delta_elec)*E_bph/E_Rph )
pf=cos(phi)
printf('power factor of the motor is %.5f lagging',
   pf)
```

Scilab code Exa 7.17 TO CALCULATE ARMATURE CURRENT DRAWN BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
clc,clear
printf('Example 7.17\n\n')

V_L=400,V_ph=V_L/sqrt(3)
E_bph=V_ph
P=4,f=50//Pole and frequency
delta_mech=4*(%pi/180) //mechanical angle in degrees
    by which rotor is behind
delta_elec= delta_mech *(P/2) //delta_mech convertd
   to electrical degrees
Z_s=complex(0,2) //synchronous impedance

//referring to phasor diagram
BC=  E_bph*sin(delta_elec)
AB=  E_bph
```

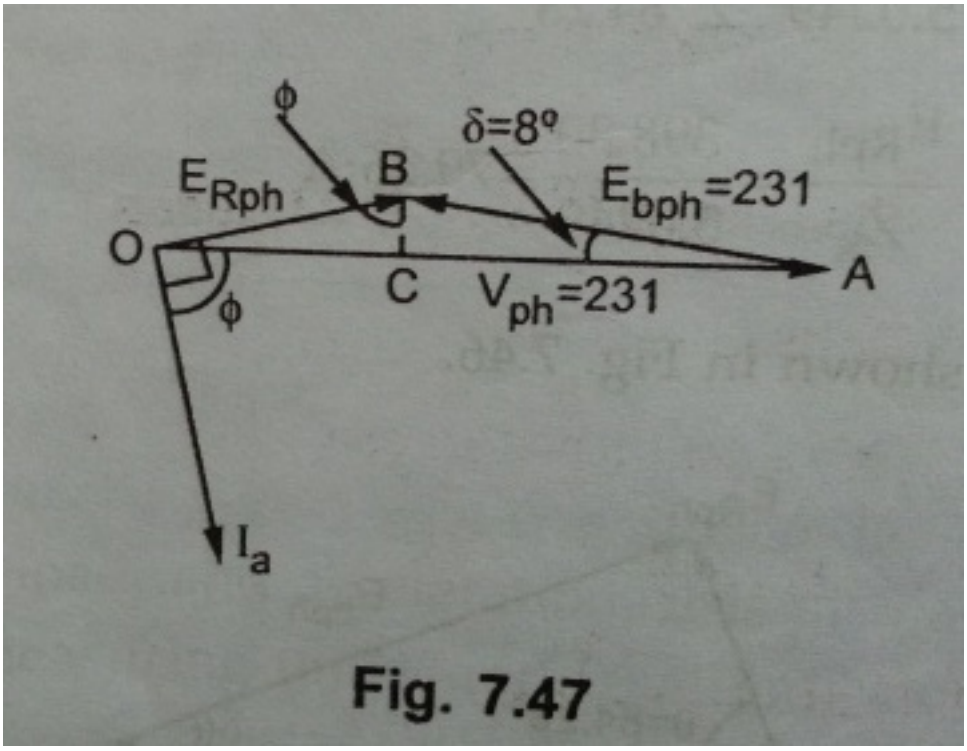


Figure 7.10: TO CALCULATE ARMATURE CURRENT DRAWN BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

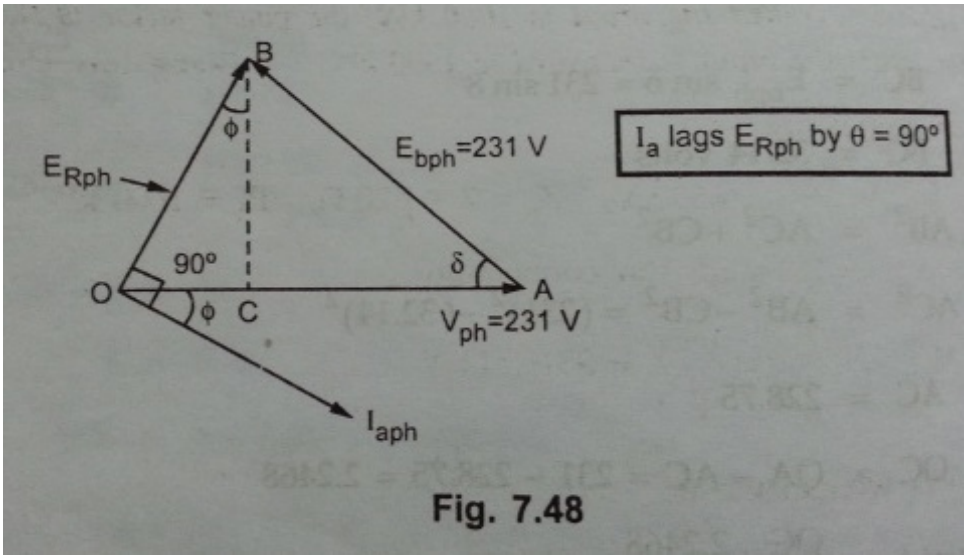


Figure 7.11: TO CALCULATE PF LOAD ANGLE AND ARMATURE CURRENT OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
14 OA=  V_ph
15
16 AC=  sqrt(AB^2-BC^2)
17 OC=  OA-AC
18 phi=atan(OC/BC)
19 OB=sqrt(OC^2 + BC^2)
20 I_a=OB/abs(Z_s)
21
22 printf('Armature current drawn by the motor is %.4f
   A',I_a)
```

Scilab code Exa 7.18 TO CALCULATE PF LOAD ANGLE AND ARMATURE CURRENT OF 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
clc,clear
printf('Example 7.18\n\n')

V_L=400,V_ph=V_L/sqrt(3)
input_power=5472
Z_s=complex(0,10)  //synchronous impedance
I_L_cosphi=input_power/(sqrt(3)*V_L) //product of
   I_L & cos(phi)
BC=10*I_L_cosphi
AB=V_ph
OA=V_ph
//from Triangle ABC in phasor diagram
AC = sqrt(AB^2- BC^2)
OC = OA - AC

//from Triangle OCB
OB=sqrt( OC^2+ BC^2 )
E_Rph = OB
I_L=E_Rph/abs(Z_s)

phi=atan(OC/BC)
pf=cos(phi)
delta=atan(BC/AC)  //load angle
printf('Power factor is %.4f lagging\n',pf)
printf('Load angle is %.0f degrees\n',delta*(180/%pi
   ))
printf('Armature current is %.3f A',I_L)
```

Scilab code Exa 7.19 TO FIND POWER FACTOR WHEN INPUT IS INCREASED

```
clc,clear
printf('Example 7.19\n\n')
```

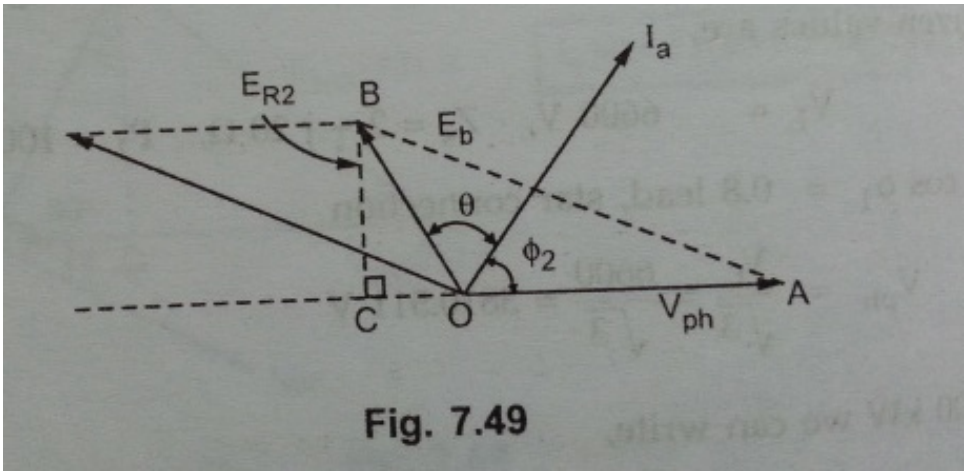


Figure 7.12: TO FIND POWER FACTOR WHEN INPUT IS INCREASED

```
3
4  V_L=6600,V_ph=V_L/sqrt(3)
5  Z_s=complex(2,20) //synchronous impedance
6  theta=(%pi/180) * phasemag(Z_s) //phasemag returns
       angle in degrees,not radians
7  P_1=1000*10^3
8  P_2=1500*10^3
9  phi_1=acos(0.8)  //leading
10
11 I_L1=P_1/(sqrt(3)*V_L*cos(phi_1))
12 I_a1ph=I_L1
13 E_R1ph=I_a1ph*abs(Z_s)
14 E_bph=  sqrt(    V_ph^2 + E_R1ph^ -2*V_ph*E_R1ph*cos
       (theta+phi_1)  )
15 I_a2_cosphi_2=P_2/(sqrt(3)*V_L)
16
17 //Refer to the phasor diagram and solving for I_y
18 //404I_y^2 -152399.968 I_y -4543000=0
19 p=[404 -152399.968 -4543000]
20 roots(p)
21 I_y=abs(ans(2)) //becuase root 1 is too high and
       root is -ve
```

```
22
23 I_a2=complex(I_a2_cosphi_2,I_y)
24 phi_2=phasemag(I_a2)
25 printf('Required power factor is %.3f leading',cosd(
   phi_2))
```

Scilab code Exa 7.20 TO DETERMINE EMF GENERATED BY 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
1 clc,clear
2 printf('Example 7.20\n\n')
3
4 V_L=2300,V_ph=V_L/sqrt(3)
5 I_L=200, I_a=I_L
6 Z_s=complex(0.2,2.2) //synchronous impedance
7 theta=(%pi/180)*phasemag(Z_s) //phasemag returns
   angle in degrees, not radians
8 phi=acos(0.5)
9
10 E_Rph=I_a*abs(Z_s)
11 E_bph =   sqrt( E_Rph^2 + V_ph^2 - 2*E_Rph*V_ph*cos(
   phi+theta)  )
12
13 printf('Generated EMF per phase is %.3f V',E_bph)
```

Scilab code Exa 7.21 TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER OF SYNCHRONOUS MOTOR

```
1 clc,clear
2 printf('Example 7.21\n\n')
```

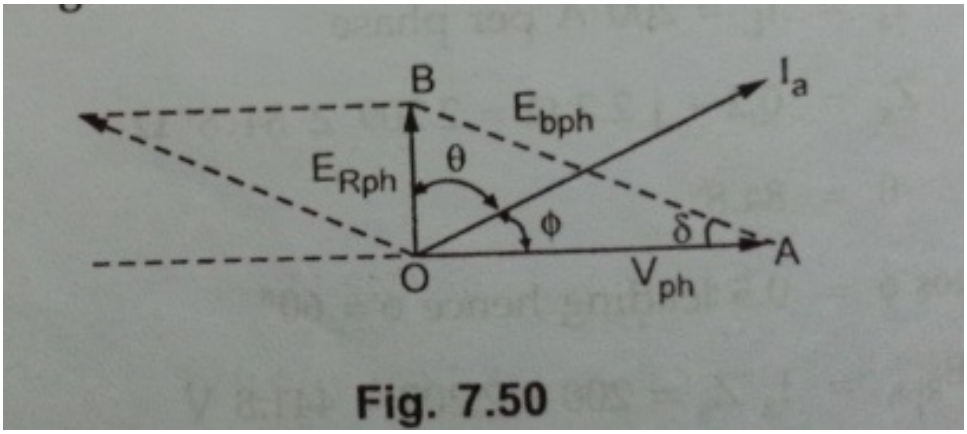


Figure 7.13: TO DETERMINE CERTAIN QUANTITIES RELATED TO MAXIMUM MECHANICAL POWER OF SYNCHRONOUS MOTOR

```
3
4  V_L=3300, V_ph=V_L/sqrt(3)
5  E_bline=3800, E_bph=E_bline/sqrt(3)
6
7  R_a=2,X_s=18 //armature resistance and synchronous
       reactance
8  Z_s=complex(R_a,X_s) //synchronous impedance
9  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
       angle in degrees,not radians
10
11 //part(i)
12 P_m_max = (E_bph*V_ph/abs(Z_s))- (E_bph^2/abs(Z_s))*
       cos(theta)  //maximum total mechanical power
13 printf('(i)Maximum total mechanical power that the
       motor can develop is %.2f W per phase',P_m_max )
14 //part(ii)
15 delta=theta //for max P_m
16 E_Rph=sqrt(  E_bph^2 + V_ph^2 -2*E_bph*V_ph*cos(
       delta) )
17 I_aph= E_Rph/abs(Z_s)
18 printf('\n(ii)Current at maximum power developed is
       %.1f A',I_aph)
```

```
19 cu_loss_total = 3*I_aph^2*R_a  //total copper loss
20 P_m_max_total=3*P_m_max //total maximum total
      mechanical power
21 P_in_total = P_m_max_total+ cu_loss_total  //total
      input power
22
23 pf=P_in_total/(sqrt(3)*V_L*I_aph)
24 printf('\n    Power factor at maximum power
      developed is %.3f leading',pf)
```

Scilab code Exa 7.22 TO DETERMINE kVA INPUT TO SYNCHRONOUS MOTOR AND ITS POWER FACTOR WHEN DRIVING 6 kW LOAD

```
1  clc,clear
2  printf('Example 7.22\n\n')
3
4  //subscript 1 refers to load 1
5  I_1=18
6  phi_1=acos(0.8)
7  V_L=440
8  S_1=sqrt(3)*I_1*V_L /1000  //kVA for load 1
9  P_1=S_1*cos(phi_1)
10 Q_1=S_1*sin(phi_1)
11
12 P_out=6
13 eta_motor=88/100
14 P_2=P_out/eta_motor
15
16 P_T=P_1+P_2
17 phi_T=acos(1)  //total power factor angle
18 Q_T=P_T*tan(phi_T)
19
20 Q_2=  Q_T - Q_1   //kVAR supplied by motor
21 //this will have  a negative sign just indicating
      its leading nature
```

```
22 phi_2=atan(abs(Q_2)/P_2)
23 pf=cos(phi_2)  //leading
24 S_2=P_2/cos(phi_2)  //kVA input to the motor
25 printf('kVA input to the motor is %.3f kVA \n',S_2)
26 printf('Power factor when driving a 6kW mechanical
      load is %.4f leading',pf)
```

Scilab code Exa 7.23 TO DETERMINE MINIMUM CURRENT AND INDUCED EMF AT FULL LOAD

```
1  clc,clear
2  printf('Example 7.23\n\n')
3
4  output_power=8*10^3
5  V_L=400,V_ph=V_L/sqrt(3)
6  R_a=0,X_s=8//armature resistance and syncronous
      reactance
7  Z_s=complex(R_a,X_s) //synchronous impedance
8  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
      angle in degrees, not radians
9  eta=88/100, input_power=output_power/eta
10
11 //minimum current occurs at max power factors
12 phi=acos(1)
13 I_a_min=input_power/(sqrt(3)*V_L*cos(phi)) //
      required minimum current
14 printf('Minimum current is %.3f A',I_a_min)
15 E_R= I_a_min * abs(Z_s)
16 E_bph =   sqrt( E_R^2 + V_ph^2 - 2*E_R*V_ph*cos(phi+
      theta)  )
17 printf('\nInduced EMF at full-load is %.3f V',E_bph)
```

Scilab code Exa 7.24 TO DETERMINE PF WHEN INPUT OF SYNCHRONOUS MOTOR IS INCREASED

```
clc,clear
printf('Example 7.24\n\n')

R_a=0.8,X_s=5
Z_s=complex(R_a,X_s) //armature resistance and
   syncronous reactance
theta=(%pi/180)*phasemag(Z_s) //synchronous
   impedance
alpha=(%pi/2) - theta
V_t=3300/sqrt(3)
P_e_in=800/(3) //per phase
phi=acos(0.8)  //leading
Q_e_in=-P_e_in*tan(phi)

// Using the following equation
// P_e_in= V_t^2*R_a/(abs(Z_s))^2   +   V_t*E_b*sin(
   delta-alpha)/abs(Z_S)
// Q_e_in= V_t^2*X_s/(abs(Z_s))^2   -   V_t*E_b*cos(
   delta-alpha)/abs(Z_S)
E_b_sin_delta_minus_9 = 407.2
E_b_cos_delta_minus_9 =2413.6
//solving further
delta = ( atand(E_b_sin_delta_minus_9/
   E_b_cos_delta_minus_9 ) + 9)
E_b=E_b_sin_delta_minus_9/sind(delta-9)

P_e_in_new = 1200*10^3/3
// Using the following equation again
// P_e_in= V_t^2*R_a/(abs(Z_s))^2   +   V_t*E_b*sin(
   delta-alpha)/abs(Z_S)
// Q_e_in= V_t^2*X_s/(abs(Z_s))^2   -   V_t*E_b*cos(
   delta-alpha)/abs(Z_S)

alpha =delta - asind(   (P_e_in_new - V_t^2*R_a/(abs
   (Z_s))^2 ) / (V_t*E_b/abs(Z_s))   )
```

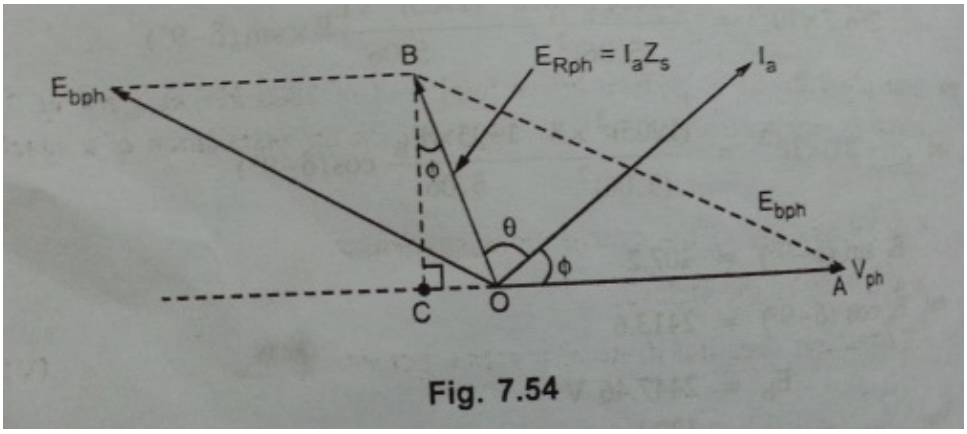


Figure 7.14: TO DETERMINE CURRENT AND PF OF A 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
28 Q_e_in_new= V_t^2*X_s/(abs(Z_s))^2   -   V_t*E_b*
      cosd(delta - alpha)/abs(Z_s)
29
30 pf=cos ( atan(abs(Q_e_in_new/P_e_in_new)))
31 printf('New power factor is %.2f leading ',pf)
```

Scilab code Exa 7.25 TO DETERMINE CURRENT AND PF OF A 3 PHASE STAR CONNECTED SYNCHRONOUS MOTOR

```
1 clc,clear
2 printf('Example 7.25\n\n')
3
4 V_L=6.6*10^3,V_ph=V_L/sqrt(3)
5 P_in=900*10^3
6 R_a=0,X_s=20 //armature resistance and synchronous
     reactance
7 Z_s=complex(R_a,X_s) //synchronous impedance
8 theta=phasemag(Z_s)*(%pi/180) //phasemag returns
```

```
    angle in degrees,not radians
E_b_L=8.6*10^3,E_bph=E_b_L/sqrt(3)

//refer to phasor diagram
OA=V_ph, AB=E_bph //OB= E_Rph

I_a_cosphi=P_in/(sqrt(3)*V_L)  //I_a*cos(phi)
BC=I_a_cosphi*abs(Z_s) //BC is a vector in phasor
    diagram

OC=sqrt(AB^2 -BC^2  )- OA  //from phasor diagram
I_a_sinphi=OC/abs(Z_s)  //product of I_a and sin(phi
    )
phi= atan (I_a_sinphi/I_a_cosphi)
I_a=I_a_cosphi/cos(phi) //product of I_a and cos(phi
    )
printf('Motor current is %.3f A\n',I_a)
printf('Power factor of motor is %f leading',cos(phi
    ))
printf('\n\nNote:There is slight mismatch in answer
    due to the approximation made during I_a* sin(phi
    ) calculation')
```

Scilab code Exa 7.26 TO DETERMINE THE kVA RATING OF SYNCHRONOUS CONDENSER USED TO IMPROVE THE PF AND THE FACTORY

```
clc,clear
printf('Example 7.26\n\n')

//subscipt 1 refers to factory load
P_1=1800
phi_1=acos(0.6) //lagging
Q_1=P_1*tan(phi_1)

```

```
9 //Subscript 2 refers to synchronous condenser
10 P_2=0
11
12 //Subscript T refers to combination of condenser and
     factory load
13 P_T=P_1+P_2
14 phi_T=acos(0.95) //lagging
15 Q_T=P_T*tan(phi_T)
16
17 kva_rating=sqrt(P_T^2+ Q_T^2)
18
19 Q_2=Q_T - Q_1
20 printf('(i)kVA rating of synchronous condender is %
     .3f kVA \n  Minus sign indicates leading nature\
     n\n',(Q_2))
21 printf('(ii)kVA rating of total factory is %.4f kVA'
     ,kva_rating)
```

Scilab code Exa 7.27 TO CALCULATE kVA INPUT AND PF OF SYNCHRONOUS MOTOR AT A CERTAIN INSTANT

```
1 clc,clear
2 printf('Example 7.27\n\n')
3
4 I_1=35
5 phi_1=acos(0.8)
6 V_L=440
7 S_1=sqrt(3)*I_1*V_L /1000 //in kVA
8
9 P_1=S_1*cos(phi_1)
10 Q_1=S_1*sin(phi_1)
11
12 P_out=12 //motor load
13 eta_motor=85/100
14 P_2=P_out/eta_motor
```

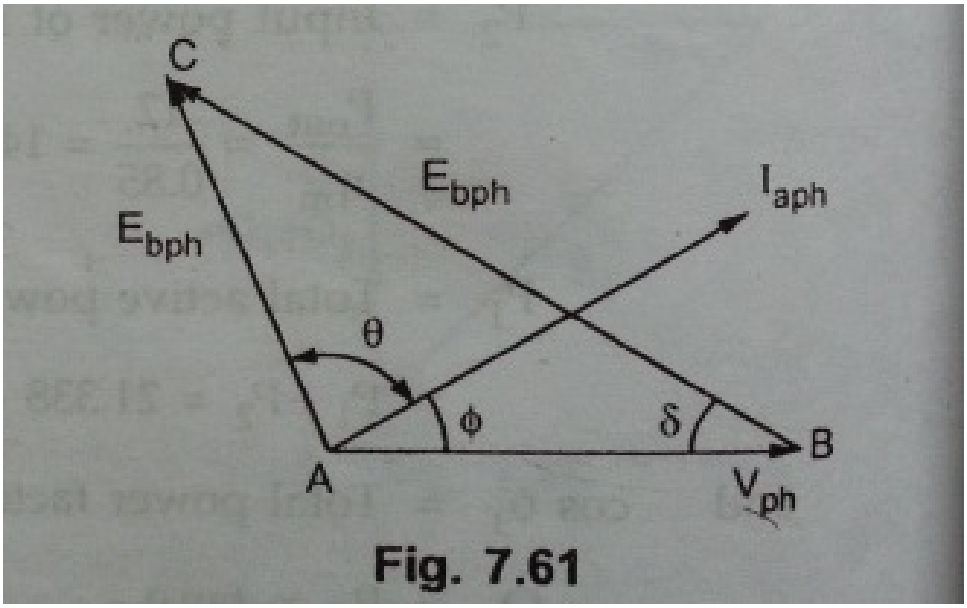


Figure 7.15: TO DETERMINE MAXIMUM OUTPUT POWER OF SYNCHRONOUS MOTOR

```
15
16 P_T=P_1 + P_2
17 phi_T=acos(1)
18 Q_T=P_T * tan(phi_T)
19
20
21 Q_2=Q_T - Q_1  //kVA supplied by motor
22 //negative sign of Q_2 indicates its leading nature
23 phi_2= atan(abs(Q_2)/P_2)
24 S_2=P_2/cos(phi_2)
25
26 printf('Power factor when motor supplies 12kW load
      is %.4f leading',cos(phi_2))
27 printf('\nkVA input to the motor is %.3f kVA',S_2)
```

Scilab code Exa 7.28 TO DETERMINE MAXIMUM OUTPUT POWER OF SYNCHRONOUS MOTOR

```
clc,clear
printf('Example 7.28\n\n')

V_L=400,V_ph=V_L/sqrt(3)
Z_s=complex(0.5,4)  //synchronous impedance
theta=(%pi/180)*phasemag(Z_s) //phasemag returns
   angle in degrees ,not radians

I_aph=60
phi=acos(0.866) //leading
power_losses=2*10^3

E_bph =   sqrt( (I_aph*abs(Z_s))^2 + (V_ph)^2 - 2*(
   I_aph*abs(Z_s))*(V_ph)*cos(phi+theta)  )
delta=theta //for P_m_max
P_m_max = (E_bph*V_ph/abs(Z_s))- (E_bph^2/abs(Z_s))*
   cos(delta)
P_m_max_total= 3 * P_m_max
P_out_max=  P_m_max_total- power_losses
printf('Maximum power output is %.4f kW',P_out_max
   *10^-3)
```

Scilab code Exa 7.29 TO DETERMINE INPUT POWER AND INDUCED EMF AT TWO DIFFERENT POWER FACTORS

```
clc,clear
printf('Example 7.29\n\n')

V_L=6.6*10^3, V_ph=V_L/sqrt(3)
```

```
I_L=50,I_aph=I_L
Z_s=complex(1.5,8) //synchronous impedance
theta=(%pi/180)*phasemag(Z_s) //phasemag returns
   angle in degrees,not radians
E_Rph=I_aph*abs(Z_s)

//part(i)
phi=acos(0.8)
P_in= sqrt(3)*V_L*I_L*cos(phi)   //for both lag and
   lead, supplied power will be the same
printf('(i)Power supplied to the motor is %.3f kW\n'
   ,P_in*10^-3)
//part(ii)
E_bph_lag =   sqrt( E_Rph^2 + V_ph^2 - 2*E_Rph*V_ph*
   cos(theta-phi)  )  //for lagging power factor
//Note that E_bph_lag > V_ph
printf('(ii)Induced EMF for 0.8 power factor lag is
   %.3f V\n',E_bph_lag)
E_bph_lead =   sqrt( E_Rph^2 + V_ph^2 - 2*E_Rph*V_ph
   *cos(theta+phi)  )  //for leading power factor
//Note that E_bph_lead < V_ph
printf('    Induced EMF for 0.8 power factor lead is
    %.3f V',E_bph_lead)
```

Scilab code Exa 7.30 TO DETERMINE AT FULLLOAD THE MINIMUM CURRENT AND ITS CORRESPONDING EMF

```
clc,clear
printf('Example 7.30\n\n')

V_L=400,V_ph=V_L/sqrt(3)
P_out=7.5*735.5
eta=85/100 //efficiency
R_a=0,X_s=10 //armature resistance and synchronous
   reactance
```

```
8  Z_s=complex(R_a,X_s)//synchronous impedance
9  theta=(%pi/180)*phasemag(Z_s) //phasemag returns
       angle in degrees,not radians
10 P_in=P_out/eta
11 phi=acos(1) //for mimimum current, power factor is
       maximum
12 I_L=P_in/(sqrt(3)*V_L*cos(phi))  , I_aph=I_L
13 printf('Minimum current is %.3f A at full load
       condition ',I_L)
14
15 E_Rph= I_aph*abs(Z_s)
16 E_bph =   sqrt( E_Rph^2 + V_ph^2 - 2*E_Rph*V_ph*cos(
       phi+theta)  )
17 printf('and corresponding EMF is %.4f V',E_bph)
```

Scilab code Exa 7.31 TO DETERMINE MAXIMUM POWER AND TORQUE A THREE PHASE SYNCHRONOUS MOTOR CAN DELIVER

```
1  clc,clear
2  printf('Example 7.31\n\n')
3
4  V_L=3.3*10^3, V_ph=V_L/sqrt(3), V_t=V_ph
5  Pole=24,f=50  //Pole and frequency
6  P=1000*10^3
7  R_a=0,X_s=3.24 //armature resistance and synchronous
       reactance
8  Z_s=complex(R_a,X_s) //synchronous impedance
9  theta=phasemag(Z_s)*(%pi/180) //phasemag returns
       angle in degrees,not radians
10 phi=acos(1)
11 I_aph=P/(sqrt(3)*V_L*cos(phi))
12
13 E_Rph=I_aph*abs(Z_s)
14 E_bph =  sqrt( E_Rph^2 + V_ph^2 - 2*E_Rph*V_ph*cos(
       phi+theta)  )
```

```
15
16 P_m_max=3*(E_bph*V_ph/abs(Z_s)) //maximum power that
       can be delivered
17 N_s=120*f/Pole //synchronous speed
18 T_max=P_m_max /(2*%pi*N_s/60) //maximum torque that
      can be developed
19 printf('Maximum power and torque the motor can
      deliver is %.3f kW and %.2f *10^3 Nm respectively
      ',P_m_max*10^-3,T_max/1000)
```