References

Aoyama, A.; Doyle, F. J.; and Venkatasubramanian, V. 1996. Control Affine Neural Network Approach for Non-Minimum Phase Nonlinear Process Control. J. Proc. Contr. 6 (1): 17-26.

Baratti, R.; Vacca, G.; and Servido, A. 1995. Neural Networks Modeling: Applications in Inferential Control Strategies. Am Inst Chem Eng Annual Nov. Meeting. Miami, FL.

Basualdo, M. S. and Ceccatto, H. A. 1995. Predictive Control Methods for Distillation Columns Using Neural Networks. DYCORD Int. Fed. Auto. Cont. (IFAC) Symp. 95: 171-176.

Beal, R. and Jackson, T. 1990. Neural Computing: An Introduction. Adam Hilger, Bristol.

Bequette, B. W. 1991. Nonlinear Control of Chemical Processes: A Review. Ind. Engng. Chem. Res. 30: 1391-1413.

Bhat, N. and McAvoy, T. J. 1989. Use of Neural Networks for Dynamic Modeling of Chemical Process Systems. IFAC Dynamics and Control of Chemical Reactors (DYCORD +'89), Maastricht, The Netherlands, 169-175.

Bhat, N. and McAvoy, T. J. 1990. Use of Neural Networks for Dynamic Modeling and Control of Chemical Process Systems. Computers chem. Engng. 14: 573-583.

Blum, J. 1992. Model Predictive Control Using Artificial Neural Networks. M.S. Thesis, University of Texas at Austin.

Blum, J.; Villard, P.; Leuba, A.; and Himmelblau, D. M. 1992. Practical issues I: Applying Neural Networks for Identification in Model Based Predictive Control. Presented at the AIChE annual Meeting, Miami

Boslovic, J. D. and Narendra, K. S. 1995. Comparison of Linear, Non-Linear and Neural-Network-Based Adaptive Controllers for a Class of Fed-Batch Fermentation Process. Automatica 31 (6): 817-840.

Brown, M. W.; Penlidis, A.; and Sullivan, G. R. 1991. Control Policies for An Industrial Acetylene Hydrogenation Reactor. The Canadian J. of Chem. Eng. 69: 152-164.

Brown, M. and Harris, C. 1994. Neurofuzzy Adaptive Modeling and Control. Englewood Cliffs, NJ: Prentice Hall.

Bulsari, A. 1995. Neural Networks for Chemical Engineers - Computer Aided Chemical Engineering Series. vol. 6. London: Elsevier.

Carpenter, G. A. and Grossberg, S. 1988. The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. Computers chem. Engng. March, 77-88.

Chen, S. and Billings, S. 1992. Neural Networks and System Identification. IEE Control Engineering Series no. 46, IEE, UK.

Chen, S.; Billings, S.; Cowan, C.; and Grant, P. 1990. Non-Linear System Identification Using Radial Basis Functions. Int. J. Syst. Sci. 21: 2513-2539.

Chen, S.; Billings, S.; Cowen, C.; and Gren, P. 1990. Practical Identification of NARMAX Models Using Radial Basis Functions. Int. J. Contr. 52: 1327-1350.

Chen, Q. and Weigand, W. A. 1994, September. Dynamic Optimization of Nonlinear Processes by Combining Neural Net Model with UDMC. AIChE J. 40 (9): 1488-1497.

Cherkassky, V.; Gehring, D.; and Mulier, F. 1995. Pragmatic Comparison of Statistical and Neural Network Methods for Function Estimation. Proc. World Cong on Neural Networks Conf., vol. 2, Washington, DC, p. 917.

Chitra, S. P. 1992, November. Neural Net Applications in Chemical Engineering. AI Expert. 20-25.

Chovan, T.; Catfolis, T.; and Meert, K. 1996. Neural Network Architecture for Process Control Based on the RTRL Algorithm. AIChE J. 42 (2): 493-502.

Cybenko, G. 1989. Approximation by Superposition of a Sigmoid Function. Math. Contr. Signals. Syst. 2: 304-314.

Dayal, B. S.; Taylor, P. A.; and MacGregor, J. F. 1994. The Design of Experiments, Training and Implementation of Nonlinear Controllers Based on Neural Network. The Canadian. J. Chem. Engng. 72: 1067-1079.

Dirion, L., et al. 1995. Design of A Neural Controller by Inverse Modeling. Computers chem. Engng. 19 (Suppl.): S797-S802.

Doherty, S. K.; Williams, D. and Gomm, J. B. 1995. Neural Network Identification and Predictive Control of an In-Line pH Process. Inst. Chem. Eng. -- Advances in Process Control 4 Meeting, 57-64.

Douglas, J. M. 1972. Process Dynamics and Control: Analysis of Dynamics Systems. vol. 1, Englewood Cliffs, NJ: Prentice Hall.

Downs, J. C. and Vogel, E. F. 1993. A Plant-Wide Industrial Process Control Problem. Computers chem. Engng. 13: 245-255.

Draeger, A.; Engell, S.; and Ranke, H. 1995. Model Predictive Control Using Neural Networks. IEEE Control Systems. 61-66.

Dutta, P. and Rhinehart, R. R. 1995. Experimental Comparison of a Novel, Simple, Neural Network Controller and Linear Model Based Controllers. Proc. Am. Cont. Conf. 1787-1789.

Economou, C. G.; Morari, M.; and Palsson, B. O. 1986. Internal Model Control. Extension to Nonlinear Systems. Ind. Engng. Chem. Process. Des. Dev. 25: 403-411.

Edward, N. J. and Goh, C. J. 1995. Direct Training Method for a Continuous Time Nonlinear Optimal Feedback Controller. J. Optimization Theory Applications 84 (3): 509-528.

Elman, J. L. 1990. Finding Structure in Time. Cognitive Science 14: 179-211.

Emmanouilides, C. and Petrou, L. 1997. Identification and Control of Anaerobic Digesters Using Adaptive, Online Trained Neural Networks. Computers chem. Engng. 21 (1): 113-143.

Evans, J. T., et al. 1993. An Online Application of a Neural Network Based Predictive Control Strategy. Proc. IEE Colloquium Nonlinear Contr. 12: 1-4.

Fan, J. Y.; Nikolaou, M.; and White, R. E. 1993, January. An Approach to Fault Diagnosis of Chemical Process via Neural Networks. AIChE J. 39 (1): 82-88.

Fakhr-Eddine, K.; Cabassud, M.; Duverneuil, P.; Le Lann, M. V.; and Couderc, J. P. 1996. Use of Neural Network for LPCVD Reactors Modeling. Computers chem. Engng. 20 (Suppl.): S521-S526.

Gokhale, V.; Horuwitz, S.; and Riggs, J. B. 1995. A Comparison of Advanced Distillation Control Techniques for a Propylene-Propane Splitter. Ind. Engng. Chem. Res. 34: 4413-4419.

Goldberg, D. 1989. Genetic Algorithms in Search. Optimization and Machine Learning, Addison-Wesley, Reading, USA.

Henson, M. A. and Seborg, D. E. 1990. Input-Output Linearization of General Nonlinear Processes. AIChE J. 36: 1753-1757.

Hernandez, E. and Arkun, Y. 1990. Neural Network Modeling and Extended DMC Algorithm to Control Nonlinear Systems. Am. Contr. Conf., Boston 1: 2454-2459.

Hecht-Nielsen, R. 1990. Neurocomputing. Addison-Wesley, Reading, USA.

Himmelblau, D. M. and Karjala, T. W. 1996. Rectification of Data in a Dynamic Process Using Artificial Neural Networks. Computers chem. Engng. 20 (6/7): 805-812.

Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, USA.

Hopfield, J. J. 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences 79: 2554-2558.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2: 359.

Huang, W. 1979. Optimize Acetylene Removal. Hydrocarbon Processing 58 (10): 131-132.

Hunt, K. J. and Sbarbaro, D. 1992. Studies in Neural Network Based Control. IEE Control Engineering Series, London.

Hunt, K. J.; Sbarbaro, D.; Zbikowski, R.; and Gawthrop, P. 1992. Neural Networks for Control Systems - A Survey. Automatica 28 (6): 1083-1112.

Hussain, M. A. 1999. Review of the Application of Neural Networks in Chemical Process Control - simulation and online implementation. Artificial Intelligence in Engineering 13: 55-68.

Hussain, M. A.; Kershenbaum, L. S. and Allwright, J. C. 1995. Online Implementation of Inverse-Model-Based Control Strategy Using Neural Networks on Partially Simulated Reactor. Am. Inst. Chem. Eng. Annual Nov. Meeting, Miami, FL.

Jordan, M. I. 1986. Attractor Dynamics and Parallelism in a Connectionist Sequential Machines. Proceedings of the 8th Annual Conference of the Cognitive Science Society, 531-546.

Joseph, B.; Wang, F. H.; and Shieh, D. S. 1992. Exploratory Data Analysis: A Comparison of Statistical Methods with Artificial Neural Networks. Computers chem. Engng. 16: 413.

Karim, M. N. and Rivera, S. L. 1992. Comparison of Feedforward and Recurrent Neural Networks for Bioprocess State Estimation. Computer chem. Engng. Suppl. S369-S377.

Karjala, T. W. and Himmelblau, D. M. 1994. Dynamic Data Reconciliation by Recurrent Neural Networks vs. Tradition Methods. AIChE J. 40: 1865-1875.

Karjala, T. W. and Himmelblau, D. M. 1996, August. Dynamic Rectification of Data via Recurrent Neural Nets and the Extended Kalman Filter. AIChE J. 42 (8): 2225-2239.

Karjala, T. W.; Himmelblau, D. M.; and Mikkulainen, R. 1992. Data Rectification Using Recurrent (Elman) Neural Networks. In Proc. Int. Joint. Conf. on Neural Networks.

Keeler, J., et al. 1997. The Process Perfector - The Next Step in Multivariable Control and Optimization. Technical Report, Pavilion Technologies Inc., Austin, TX.

Kershenbaum, L. and Kittisupakorn, P. 1994. The Use of a Partially Simulated Exothermic Reactor for Experimental Testing of Control Algorithm. Chemical Engineering Research and Design: Process Operation and Control (January) pp. 55-67.

Khalid, M. and Omatu, S. 1992. A Neural Network Controller for a Temperature Control System. IEEE Contr. Syst. Mag. 58-64.

Khalid, M.; Omatu, S.; and Yusuf, R. 1993. MIMO Furnace Control with Neural Networks. IEEE Trans. Contr. Syst. Technol. 1 (4): 238-245.

Kittisupakorn, P. and Kershenbaum, L. 1994. Use of Predictive Control Techniques for the Control of a Reactor with Exothermic Reactions. The 1994 ICHEME Research Event, London, England (October) pp. 979-983.

Kohonen, T. 1989. Self-Organizing and Associative Memory (3rd ed.). Springer - Verlag, Berlin, Germany.

Kramer, M. A. 1992. Autoassociative Neural Networks. Computers chem. Engng. 16: 313.

Kramer, M. A. and Leonard, J. A. 1990. Diagnosis Using Neural Networks - Analysis and Criticism. Computers chem. Engng. 14: 1323-1338.

Krothapally, K. and Palanki, S. 1997. A Neural Network Strategy for Batch Process Optimization. Computers chem. Engng. 21 (Suppl.): S463-S468.

Kurtanjek, Z. 1994. Modeling and Control by Artificial Neural Network in Biotechnology. Computers chem. Engng. 18 (Suppl.): S627-S631.

Lambert, J. and Hecht-Nielson, R. 1991. Application of Feedforward and Recurrent Neural Networks to Chemical Plant Predictive Modeling. In Proc. Int. Joint Conf. on Neural Networks 1: 373-379.

Lam and Lloyd. 1992, March. Catalyst Aids Selective Hydrogenation of Acetylene. Oil & Gas J. 27.

Langonet, P. 1993. Neural Nets for Process Control. Conf. on Precision Processing Technology, Netherlands. 631-640.

Lee, M. and Park, S. 1992. A New Scheme Combining Neural Feedforward Control with Model Predictive Control. AIChE J. 38 (2): 193-200.

Lee, P. L. and Sullivan, G. R. 1988. Generic Model Control. Computers chem. Engng. 12: 573-598.

Leonard, J. and Kramer, M. A. 1990. Improvement of the Backpropagation Algorithm for Training Neural Networks. Computers chem. Engng. 14 (3): 337-341.

Lightbody, C., et al. 1992. Control Applications for Feedforward Networks. IEE Control Engineering Series no. 46. IEE, UK 51-71.

Lightbody, G. and Irwin, G. W. 1995. A Novel Neural Internal Model Control Structure. Proc. Am. Contr. Conf. 350-354.

Lightbody, G. and Irwin, G. W. 1995. Direct Neural Model Reference Adaptive Control. IEE Proc. -- Contr. Theory Applications 142 (1): 31-43.

Linkens, D. A. and Nie, J. 1994. Backpropagation Neural - Network Based Fuzzy Controller with a Self-Tuning Teacher. Int. J. Contr. 60 (1): 17-39.

Lippmann, R. P. 1987. An Introduction to Computing with Neural Nets. IEE ASSP Mag. 4: 4-22.

Ljung, L. 1987. System Identification: Theory for the User. Englewood Cliffs, NJ: Prentice-Hall.

Loh, A. P.; Looi, K. O.; and Fong, K. F. 1995. Neural Network Modeling and Control Strategies for a pH Process. J. Proc. Contr. 5 (6): 355-362.

Lou, K-N and Perez, R. A. 1996. A New System Identification Technique Using Kalman Filtering and Multilayer Neural Networks. Artificial Intelligence in Engineering. 10: 1-8.

Luyben, W. L. 1973. Process Modeling, Simulation, and Control for Chemical Engineers. New York: McGraw Hill.

MacMurray, J. C. and Himmelblau, D. M. 1995. Modeling and Control of Paced Distillation Column Using Artificial Neural Networks. Computers chem. Engng. 19 (10): 1077-1088.

McCulloch, W. S. and Pitts, W. A. 1943. A Logical Calculus of the Ideas Imminent in Nervous Activity. Bulletin of math. Biophysics 5, 115-133.

Miller, W. T.; Sutton, R. S.; and Werbos, P. J. editors. Neural Networks for Control. The MIT Press, Cambridge, MA.

Minsky, M. and Papert, S. 1969. Perceptron. The MIT Press, Cambridge, MA.

Morris, A. J.; Montague, G. A.; and Willis, M. J. 1994. Artificial Neural Networks: Studies in Process Modeling and Control. Trans. Inst. Chem. Engng. UK - Part A 72: 3-19.

Nahas, E. P.; Henson, M. A.; and Seborg, D. E. 1992. Nonlinear Internal Model Control Strategy for Neural Network Models. Computers chem. Engng. 16 (2): 1039-1057.

Naidu, S. R.; Zafiriou, E.; and McAvoy, T. J. 1990. Use of Neural Networks for Sensor Failure Detection in a Control System. IEEE Cont. Syst. Mag. 10 (3): 49-55.

Narendra, K. S. and Parthasarathy, K. 1990. Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. Neural Networks. 1: 4-27.

Näsi, M.; Alikoski, M.; and White, C. 1985. Advanced Control of Acetylene Hydrogenation Reactors. Hydrocarbon Processing. 64 (6): 57.

Nikolaou, M. and Hanagandi, V. 1993. Control of Nonlinear Dynamical Systems Modeled by Recurrent Neural Networks. AIChE J. 39 (11): 1890-1894.

Nikravesh, M.; Farell, A.; and Standford, T. G. 1996. Model Identification of Nonlinear Time Variant Processes via Artificial Neural Network. Computers chem. Engng. 20 (11): 1277-1290.

Noriega, J. R. and Wang, H. A. 1995. Direct Adaptive Neural Network Control for Unknown Nonlinear Systems and Its Application. Proc. Am. Control Conf. 4285-4289.

Pao, Y. H.; Phillips, S. M.; and Sobajic, D. J. 1992. Neural Net Computing and the Intelligent Control of Systems. Int. J. Contr. 56 (2): 263-289.

Perry, J. H. editor. 1974. The Chemical Engineer 's Handbook. New York: McGraw-Hill.

Pham, D. T. and Liu, X. 1993. Identification of Linear Systems Using Recurrent Neural Networks. In applications of neural networks to modeling and control, ed. G. F. Page, Gomm J. B. and Williams D. Chapman and Hall, London, 24-34.

Piovoso, M. J. and Owens, A. J. 1991. Sensor Data Analysis Using Neural Networks. Proc. CPC-IV Int. Conf. on Chemical Process Control, Padre Island, TX, 101.

Piovoso, M. J., et al. 1992. A Comparison of Three Nonlinear Controller Designs Applied to a Non-Adiabatic First-Order Exothermic Reaction in a CSTR. Proc. Am. Contr. Conf. 490-493.

Piron, E.; Latrille, E.; and René, F. 1997. Application of Artificial Neural Networks for Cross Flow Microfiltration Modeling: "Black-Box" and "Semi-Physical" Approaches. Computers chem. Engng. 21 (9): 1021-1030.

Pollard, J. F.; Broussard, M. R.; Garrison, D. B.; and San, K. Y. 1992. Process Identification Using Neural Networks. Computers chem. Engng. 16: 253-270.

Pottmann, M. and Seborg, D. E. 1997. A Nonlinear Predictive Control Strategy Based on Radial Basis Function Models. Computers chem. Engng. 21 (9): 965-980.

Psaltis, D.; Sideris, A.; and Yamamura, A. A. 1988. A Multilayer Neural Network Controller. IEEE Cont. Syst. Mag. 8: 17-21.

Psichogios, D. M. and Ungar, L. H. 1991. Direct and Indirect Model Based Control Using Artificial Neural Networks. Ind. Engng. Chem. Res. 30: 2564-2573.

Psichogios, D. M. and Ungar, L. H. 1992, October. A Hybrid Neural Network - First Principles Approach to Process Modeling. AIChE J. 38 (10): 1499-1511.

Ramasamy, S. and Deshpande, P. B. 1995. Process Identification and Advanced Controls with Neural Networks. Dissertation, University of Louisville, KY, USA.

Ramchandran, S. and Rhinehart, R. R. 1995. A Very Simple Structure for Neural Network Control of Distillation. J. Proc. Contr. 5 (2): 115-128.

Rumelhart, D. E.; Hinton, G. E.; and William, R. J. 1986a. Learning Internal Representations by Error Backpropagation: Parallel Distributed Processing. Vol. 1: Foundations. pp. 318-362. Ed. Rumelhart, D. E. and McClelland, J. and the PDP research group. The MIT Press, Cambridge, MA.

Rumelhart, D. E.; Hinton, G. E.; and William, R. J. 1986b. Learning Representations by Error Backpropagation. Nature 323: 225-228.

Rumelhart, D. and McClelland, J. 1986. Parallel Distributed Processing: Exploitations in the Microstructure of Cognition, Volume 1 and 2. Cambridge: MIT Press, USA.

Sabharwal, A.; Bhat, N. V.; and Wada, T. 1997, October. Integrate Empirical and Physical Modeling. Hydrocarbon Processing 105-110.

Saint-Donat, A.; Bhat, N.; and McAvoy, T. J. 1991. Neural Net Based Predictive Control. Int. J. Contr. 54: 1453-1468.

Sanner, R. M. and Slotine, J. J. 1992. Gaussian Networks for Direct Adaptive Control. IEEE Trans. Neural Networks. 3 (6): 1299-1317.

Sbarbaro, H. D.; Neumerkel, D.; and Hunt, K. 1993. Neural Control of a Steel Rolling Mill. IEEE Contr. Syst. 13 (3): 631-640.

Schbib, N. S.; Errazu, A. F.; Romagnoli, J. A.; and Porras, J. A. 1994. Dynamics and Control of an Industrial Front-End Acetylene Converter. Computers chem. Engng. 18: S355.

Scheneker, B. and Agarwal, M. 1994. Experimental Application of Control Based on State Feedback Neural Network Predictors. Am. Inst. Chem. Eng. Annual Nov. Meeting, San Francisco, CA.

Scott, G. M. and Ray, W. H. 1993. Experiences with Model-Based Controllers Based on Neural Network Process Models. J. Proc. Contr. 3 (3): 179-196.

Seborg, D. E. 1994. Experience with Nonlinear Control and Identification Strategies. IEE Control 94 Conf. 879-886.

Shah, M. A. and Meckl, P. H. 1995. On-line Control of a Nonlinear System Using Radial Basis Function Neural Networks. Proc. Am. Contr. Conf. 4265-4269.

Sheppard, C. P.; Grant, C. R.; and Ward, R. M. 1992. A Neural Network Based Furnace Control System. Proc. Am. Cont. Conf. 500-504.

Stephanopoulos, G. 1984. Chemical Process Control - An Introduction to Theory and Practice. Englewood Cliffs, NJ: Prentice Hall.

Su, H. T. and McAvoy, T. J. 1992. Long-Term Predictions of Chemical Processes Using Recurrent Neural Networks: A Parallel Training Approach. Ind. Engng. Chem. Res. 31 (5): 1338-1352.

Syu, M. and Chang, J. B. 1997. Recurrent Backpropagation Neural Network Adaptive Control of Penicillin Acylase Fermentation by Arthrobacter Viscous. Ind. Engng. Chem. Res. 36: 3756-3761.

Tan, Y. and VanCauwenberghe, A. R. 1996. Optimization Techniques for the Design of a Neural Predictive Controller. Neurocomputing. 10: 83-96.

Temeng, K. O.; Schnelle, P. D.; and McAvoy, T. J. 1995. Model Predictive Control of an Industrial Packed Bed Reactor Using Neural Networks. J. Proc. Contr. 5 (1): 19-27.

Thibault, J. and Grandjean, B. P. A. 1991. Neural Networks in Process Control - A Survey. IFAC Advanced Control of Chemical Processes, Toulouse, France.

Thompson, M. L. and Kramer, M. A. 1994. Modeling Chemical Processes Using Prior Knowledge and Neural Networks. AIChE J. 40 (8): 1328-1340.

Tsen, A. Y., et al. 1996. Predictive Control of Quality in Batch Polymerization Using Hybrid ANN Models. AIChE J. 42 (2): 455-465.

Turner, P.; Montague, G. A.; and Morris, A. J. 1995. Neural Networks in Dynamic Process State Estimation and Nonlinear Predictive Control. Fourth Int. Conf. Artificial Neural Networks, Cambridge 284-289.

Ungar, L. H.; Powell, B. A.; and Kamens, S. N. 1990. Adaptive Networks for Fault Diagnosis and Control. Computers chem. Engng. 14 (4/5): 561-572.

VanCan, H. J., et al. 1995. Design and Real Time Testing of a Neural Model Predictive Controller for a Nonlinear System. Chem. Engng. Sci. 50 (15): 2419-2430.

Venkatasubramanian, V. and Chan, K. 1989. A Neural Network Methodology for Process Fault Diagnosis. AIChE J. 35 (12): 1993-2002.

Vora, N.; Tambe, S. S.; and Kulkarni, B. D. 1997. Counterpropagation Neural Networks for Fault Detection and Diagnosis. Computers chem. Engng. 21 (2): 177-185.

Watanabe, N. A. 1994. Comparison of Neural Network Based Control Strategies for a CSTR. Adv. Cont. Chem. Process (ADCHEM) 94 Symp. 391-396.

Weigand, A. S.; Rumelhart, D. E.; and Huberman, B. A. 1990. Backpropagation, Weight Elimination and Time Series Prediction. Proc. of Summer School, Colarado. 105-116.

Weiss, G. 1996. Modeling and Control of an Acetylene Converter. J. Proc. Contr. 16 (1): 7.

Werbos, P. J. 1974. Beyond Regression: New Tools for the Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University.

Widrow, B. and Hoff, M. E. 1960. Adaptive Switching Circuits. Proc. 1960 IRE WESCON Convention Record, Part 4. IRE, New York, USA 96-104.

Widrow, B.; Rumelhart, D. E.; and Lehr, M. A. 1994. Neural Networks: Applications in Industry, Business, and Science. Comm. of the ACM. 37: 93.

Willis, M. J. et al. 1991. Artificial Neural Networks in Process Engineering. IEE Proc. - Part D. 138: 256-266.

Willis, M. J.; Montague, G. A.; and Morris, A. J., et al. 1992. Artificial Neural Networks in Process Estimation and Control. Automatica 28 (6): 1181-1187.

Witoon Suewatanakul. 1993. A Comparison of Fault Detection and Classification Using Artificial Neural Networks with Traditional Methods. Dissertation, The University of Texas at Austin.

Wormsley, C. and Henry, J. 1994. A Study of Model Predictive Control for a Distillation Column. Am. Inst. Chem. Eng. Annual Nov. Meeting, San Francisco, CA.

Yang, Y. Y. and Linkens, D. A. 1994. Adaptive Neural-Network-Based Approach for the Control of Continuous Stirred Tank Reactor. IEE Proc. - Contr. Theory Applications. 141 (5): 341-349.

Ydstie, B. E. 1990. Forecasting and Control Using Adaptive Connectionist Networks. Computers chem. Engng. 14: 583-599.

You, Y. and Nikolaou, M. 1993, October. Dynamic Process Modeling with Recurrent Neural Networks. AIChE J. 39 (10): 1654-1667.

Zurada, J. M. 1992. Introduction to Artificial Neural Systems. West Publishing, St. Paul, Minnesota.

Appendix A

Neural Network Toolbox

MATLAB abbreviated from MATrix LABoratory is a technical computing environment for high-performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment where problems and solutions are expressed just as they are written mathematically - without traditional program meaning.

MATLAB 's funtionality and versatility with the addition of optional application-specific toolboxes can be extended. Toolboxes are comprised of suites of MATLAB functions (M-files) written by world-class authorities on each of the respective topics. These toolboxes cover a variety of disciplines as illustrated by the following list

- Matlab Toolbox
- Control System Toolbox
- Signal Processing Toolbox
- System Identification Toolbox
- Optimization Toolbox
- Neural Network Toolbox
- Fuzzy Logic Toolbox
- Model Predictive Control.

A.1 Neural Network Toolbox

In *Neural Network Toolbox*, it provides many useful toolbox functions supporting programmers to do their programs easier. Four main toolbox functions used for neural network programming are as follow

- Network creation functions
- Weight/bias initialization functions
- Training functions
- Performance functions

A.2 Training Functions

Gradient descent and gradient descent with momentum are backpropagation training algorithm. However, these two methods are often too slow for practical problems. Consequently, high performance algorithms that can converge from ten to one hundred times faster than those two algorithms.

These faster algorithms fall into two main categories. The first category uses heuristic techniques, which were developed from an analysis of the performance of the standard steepest descent algorithm. One heuristic modification is the momentum technique. The others, which are more heuristic technique, are variable learning rate backpropagation, **TRAINGDA**, and resilient backpropagation, **TRAINRP**.

The second category of fast algorithms uses standard numerical optimization techniques. Three types of numerical optimization techniques for neural network training: conjugate gradient (**TRAINCGF**, **TRAINCGP**, **TRAINCGB**, **TRAINSCG**), quasi-Newton (**TRAINBFG**, **TRAINOSS**), and Levenberge-Marquardt (**TRAINLM**) are provided.

A.2.1 Gradient Descent Learning Rule

1) Variable Learning Rate (TRAINGDA, TRAINGDX)

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate change during the training process, as the algorithm moves across the performance surface.

The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface.

2) Resilient Backpropagation (TRAINRP)

Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called *squashing* functions, since they compress an infinite input range into a finite output range, Sigmoid functions are characterized by the fact that their slope must approach zero as the inputs get large. This causes a problem when using steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude, and therefore cause small changes in the weights and biases, even though the weights and biases are far from their optimal values.

The purpose of the resilient backpropagation (Rprop) training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value.

Rprop is generally much faster than the standard steepest descent algorithm. It also has the nice property that it requires only a modest increase in memory requirements. We do need to store the update values for each weight and bias, which is equivalent to storage of the gradient.

A.2.2 Conjugate Gradient Algorithms

The basic backpropagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along the conjugate directions, which produces generally faster convergence than steepest descent directions. In this section, four different variations of conjugate gradient algorithm are presented.

1) Fletcher-Reeves Update (TRAINCGF)

All of the conjugate gradient algorithms start out by searching in the steepest descent direction (negative of the gradient) on the first iteration.

$$p_0 = -g_0$$

A line search is then performed to determine the optimal distance along the current search direction:

$$x_{k+1} = x_k + \alpha_k p_k$$

Then the next search direction is determined so that it is conjugate to various search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search directions

$$p_k = -g_k + \beta_k p_{k-1}$$

The various versions of conjugate gradient are distinguished by the researcher in which the constant β_k is computed. For the Fletcher-Reeves update procedure is

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$

This is the ratio of the norm squared of the current gradient to the mean squared of the previous gradient.

2) **Polak-Ribiére Update (TRAINCGP)**

Another version of the conjugate gradient algorithm was proposed by Polak and Ribire. As with the Fletcher-Reeves algorithm, the search direction at each iteration is determined by

$$p_k = -g_k + \beta_k p_{k-1}$$

For the Polak-Ribiére Update, the constant β_k is computed by

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}$$

This is the inner product of the previous change in the gradient with the current gradient divided by the norm squared of the previous gradient. The **TRAINCGP** routine has performance similar to **TRAINCGF**. It is difficult to predict which algorithm will perform best on a given problem. The storage requirements for Polak-Ribiére (four vectors) are slightly larger than for Fletcher-Reeves (three vectors).

3) Powell-Beale Restarts (TRAINCGB)

For all conjugate gradient algorithms, the search direction will be periodically reset to the negative of the gradient. The standard reset point occurs when the number of iterations is equal to the number of network parameters (weights and biases), but there are other reset methods, which can improve the efficiency of training. One of such reset methods was proposed by Powell, based on an earlier version proposed by Beale. For this technique we will restart if there is very little orthogonality left between the current gradient and the previous gradient. This is tested with the following inequality:

$$\left|g_{k-1}^{T} g_{k}\right| \geq 0.2\left\|g_{k}\right\|^{2}$$

If this condition is satisfied, the search direction is reset to the negative of the gradient. The **TRAINCGB** routine has performance, which is somewhat better than **TRAINCGP** for some problems, although performance on any given problem is difficult to predict. The storage requirements for the Powell-Beale algorithm (six vectors) are slightly larger than for Polak-Ribiére (four vectors).

4) Scaled Conjugate Gradient (TRAINSCG)

Each of the conjugate gradient algorithms, which we have discussed so far, requires a line search at each iteration. This line search is computationally expensive, since it requires that the network response to all training inputs be computed several times for each search. The scaled conjugate gradient algorithm (SCG) was designed to avoid the time consuming line search. This algorithm is too complex to explain in a few lines, but the basic idea is to combine the model-trust region approach, which is used in the Levenberge-Marquardt algorithm describe later, with the conjugate gradient approach. The **TRAINSCG** routine may require more iteration to converge than the other conjugate gradient algorithms, but the number of computations in each iteration is significantly reduced because no line search is performed. The storage requirements for the scaled conjugate gradient algorithm are about the same as those of Fletcher-Reeves.

A.2.3 Quasi-Newton Algorithms

1) BFGS Algorithm (TRAINBFG)

Newton's method is an alternative to the conjugate gradient methods for fast optimization. The basic step of Newton's method is

$$x_{k+1} = x_k - A_k^{-1} g_k,$$

where A_k is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method often converges faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feedforward neural networks. There is a class of algorithms that are based on Newton's method but which do not require calculation of second derivatives. These are called quasi-Newton (or secant) methods. They update an approximate Hessian matrix at each iteration of the algorithm. The update is computed as a function of the gradient. The algorithm has been implemented in the **TRAINBFG** routine.

2) One Step Secant Algorithm (TRAINOSS)

Since the BFGS algorithm requires more storage and computation in each iteration than the conjugate gradient algorithms, there is need for a secant approximation with smaller storage and computation requirements. The one step secant (OSS) method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse.

A.2.4 Levenberge-Marquardt (TRAINLM)

Like the quasi-Newton methods, the Levenberge-Marquardt algorithm was designed to approach second order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$H = J^T J$$

and the gradient can be computed as

$$g = J^T e$$

where J is the Jacobian matrix, which contain first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard bachpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberge-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton like update:

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e.$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton 's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton 's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

A.3 Speed and Memory Comparison of Training Functions

It is very difficult to know which training algorithm will be the fastest for a given problem. It will depend on many factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, and the error goal. In general, on networks which contain up to a few hundred weights the Levenberge-Marquardt algorithm will have the fastest convergence. This advantage is especially noticeable if very accurate training is required. The quasi-Newton methods are often the next fastest algorithms on networks of moderate size. The BFGS algorithm does require storage of the approximate Hessian matrix, but is generally faster than the conjugate gradient algorithms. Of the conjugate gradient algorithms, the Powell-Beale procedure requires the most storage, but usually has the fastest convergence. Rprop and the scaled conjugate gradient algorithm do not require a line search and have small storage requirements. They are reasonably fast, and are very useful for large problems. The variable learning rate algorithm is usually much slower than the other method, and has about the same storage requirements as Rprop, but it can still be useful for some problems.

For most situations, Levenberge-Marquardt algorithm is recommended to try first. If this algorithm requires too much memory, then try the BFGS algorithm, or one of the conjugate gradient methods. The Rprop algorithm is also very fast, and has relatively small memory requirements.

The following table gives some example convergence times for the various algorithms on one particular regression problem. In this problem a network with one input node, ten hidden nodes, and one output node was trained on a data set with 41 input/output pairs until a mean square error performance of 0.01 was obtained. Twenty different test runs were made for each training algorithm on a Macintosh Powerbook 1400 to obtain the average numbers shown in the table. These numbers should be used with caution, since the performance shown here may not be typical for these algorithms on other type of problems. (You may notice that there is not a clear relationship between the number of floating point operations and the time required to reach convergence. This is because some of the algorithms can take advantage of efficient built-in MATLAB functions. This is especially true for the Levenberge-Marquardt algorithm.)

For most situations, Levenberge-Marquardt algorithm is recommended to try first. If this algorithm requires too much memory, then try the BFGS algorithm **TRAINBFG**, or one of the conjugate gradient methods. The Rprop algorithm **TRAINRP** is also very fast, and has relatively small memory requirements.

Table A.1: Speed and memory comparison of training functions

Function	Technique	Time	Epochs	Mflops
Traingdx	Variable Learning Rate	57.71	980	2.50
Trainrp	Rprop	12.95	185	0.56
Trainscg	Scaled Conj. Grad.	16.06	106	0.70
Traincgf	Fletcher-Powell CG	16.40	81	0.99
Traincgp	Polak-Ribiére CG	19.16	89	0.75
Traincgb	Powell-Beale CG	15.03	74	0.59
Trainoss	One-Step-Secant	18.46	101	0.75
Trainbfg	BFGS quasi-Newton	10.86	44	1.02
Trainlm	Levenberge-Marquardt	1.87	6	0.46

Appendix B

Backpropagation Algorithm

B.1 Conclusion of the Backpropagation Algorithm

- **Weight Initialization**

  Set all weights and node thresholds to small random numbers. Note that the node threshold is the negative of the weight from the bias unit (whose activation level is fixed at 1).

- **Calculation of activation function**
  1. The activation level of an input unit is determined by the instance presented to the network.
  2. The activation level O_j of a hidden and output unit is determined by

     $$O_j = F\left(\sum W_{ji} O_i + \theta_j\right)$$

     where W_{ji} is the weight from an input O_i, θ_j is the node threshold, and F is the sigmoid function:

     $$F(a) = \frac{1}{1+e^{-a}}$$

- **Weight Training**
  1. Start at the output units and work backward to the hidden layers recursively. Adjust weights by

     $$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}$$

where $W_{ji}(t)$ is the weight from unit i to unit j at time t (or t the iteration) and ΔW_{ji} is the weight adjustment.

2. The weight change is computed by

$$\Delta W_{ji} = \eta \delta_j O_i$$

where η is a trial-independent learning rate ($0<\eta<1$, e.g., 0.3) and δ_j is the error gradient at unit j. Convergence is sometimes faster by adding a momentum term:

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_j O_i + \alpha\left[W_{ji}(t) - W_{ji}(t-1)\right]$$

where $0<\alpha<1$.

3. The error gradient is given by:

   - For the output units:

   $$\delta_j = O_j(1-O_j)(T_j - O_j)$$

   where T_j is the desired (target) output activation and O_j is the actual output activation at output unit j.

   - For the hidden unit

   $$\delta_j = O_j(1-O_j)\sum_k \delta_k W_{kj}$$

   where δ_k is the error gradient at unit k to which a connection points from hidden unit j.

4. Repeat iterations until convergence in terms of the selected error criterion. An iteration includes presenting an instance, calculating activations, and modifying weights.

The name "backpropagation" comes from the fact that the error (gradient) of hidden units are derived from propagating backward the errors associated with output units since the target values for the hidden units are not given. In the backpropagation network, the activation function chosen is the sigmoid function, which compresses the output value into the range between 0 and 1. The sigmoid function is advantageous in that it can accommodate large signals without saturation while allows the passing of small signals without excessive attenuation. Also, it is a smooth function so that gradients can be calculated, which are required for a gradient descent search.

B.2 Example of Calculation

To solve the exclusive-or problem, we build a backpropagation network as shown in Figure B.1. The network will be trained on the following instances:

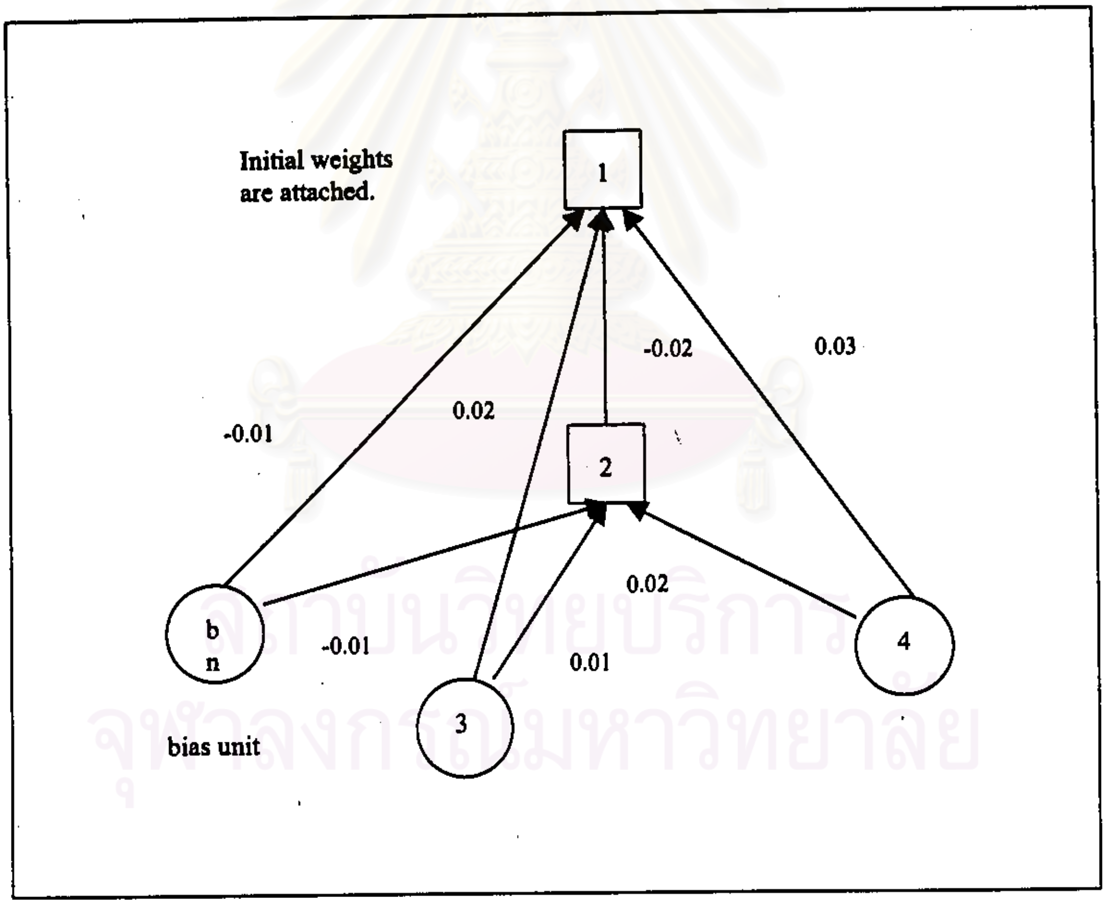

Figure B.1: A backprpagation network for learning the exclusive-or function.

Inputs	Output
(1,1)	0
(1,0)	1
(0,1)	1
(0,0)	0

The weights are initialized randomly as follows:

$$W_{13} = 0.02,\ W_{14} = 0.03,\ W_{12} = \text{-}0.02,\ W_{23} = 0.01,$$

$$W_{24} = 0.02,\ W_{1b} = \text{-}0.01,\ W_{2b} = \text{-}0.01$$

Calculation of Activation: Consider a training instance with the input vector = (1,1) and the desired output vector = (0).

$$O_3 = O_4 = 1$$

$$O_2 = 1/\left[1 + e^{-(1\times 0.1+1\times 0.2-1\times 0.1)}\right] = 0.505$$

$$O_1 = 1/\left[1 + e^{-(0.505\times -0.02+1\times 0.02+1\times 0.03-1\times 0.01)} = 0.508$$

Weight Training: Assume that the learning rate $\eta = 0.3$.

$$\delta_1 = 0.508(1-0.508)(0-0.508) = -0.127$$

$$\Delta W_{13} = 0.3\times(-0.127)\times 1 = -0.038$$

$$\delta_2 = 0.505(1-0.505)(-0.127\times -0.02) = 0.0006$$

The rest of the weight adjustments are omitted. Note that the threshold (which is the negative of the weight from the bias unit) is adjusted likewise. It takes many iterations like this before the learning (training) process stops. The following set of the final weight gives the mean squared error of less that 0.01: $\Delta W_{23} = 0.3 \times 0.0006 \times 1 = 0.002$

$W_{13} = 4.98$, $W_{14} = 4.98$, $W_{12} = -11.30$, $W_{23} = 5.62$,
$W_{24} = 5.62$, $W_{1b} = -2.16$, $W_{2b} = -8.83$

Backpropagation has been applied to classification tasks, speech synthesis from text (e.g., NETtalk), adaptive robotic control, scoring of bank loan applications, system modeling, data compression, and many others.

Appendix C

Signal Processing and Data Filtering

In process control, the noise associated with analog signals can arise from a number of sources: the measurement device, electrical equipment, or the process itself. The effects of electrically generated noise can be minimized by following established procedures concerning shielding of cables, grounding, etc. Process included noise can arise from variation due to mixing, turbulence, and non-uniform multiphase flows. The effects of both process noises can be reduced by signal conditioning or filtering.

C.1 Analog Filters

Analog filters have been used for many years to smooth noisy experimental data. For example, an *exponential filter* can be used to damp out high-frequency fluctuations due to electrical noise; hence it is called a low-pass filter. Its operation can be described by a first-order transfer function or equivalently a first-order differential equation.

$$\tau_F \frac{dy(t)}{dt} + y(t) = x(t) \qquad \text{(C.1)}$$

where x is the measured value (the filter input), y is the filtered value (the filter output), and τ_F is the time constant of the filter. Note that the filter has a steady-state gain of one. The exponential filter is also called and *RC filter* since it can be constructed from a simple RC electrical circuit.

The filter time constant, τ_F should be much smaller than the dominant time constant of the process τ_{max} to avoid introducing a significant dynamic lag in the feedback control loop. For example, choosing $\tau_F < 0.1\tau_{\max}$ satisfies this requirement. On the other hand, if the noise amplitude is high, then a larger value of τ_F may be required to *smooth* the noisy measurements. The frequency range of the noise is another important consideration. Suppose that the lowest noise frequency expected is denoted by ω_N. Then τ_F should be selected so that $\omega_F < \omega_N$ where $\omega_F = 1/\tau_F$.

C.2 Digital Filters

In this section several popular digital filters are considered. They are exponential filter, double exponential filter, moving average filter, and noise-spike filter. The description and the expression of each filter are provided respectively as follows.

C.2.1 Exponential Filter

First a digital version of the exponential filter is considered. The samples of the measured variable will be denoted as $x_{n-1}, x_{n..}$.... and the corresponding filtered values will be denoted as y_{n-1}, y_n....where n refers to the current sampling instant. The derivative in Equation (C.1) at time step n can be approximated by the backward difference:

$$\frac{dy}{dt} \cong \frac{y_n - y_{n-1}}{\Delta t} \tag{C.2}$$

Substituting in Equation (C.1) and replacing $y(t)$ by y_n and $x(t)$ by x_n yields

$$\tau_F \frac{y_n - y_{n-1}}{\Delta t} + y_n = x_n \tag{C.3}$$

Rearranging gives

$$y_n = \frac{\Delta t}{\tau_F + \Delta t} x_n + \frac{\Delta t}{\tau_F + \Delta t} y_{n-1} \tag{C.4}$$

Define

$$\alpha = \frac{1}{\tau_F / \Delta t + 1} \tag{C.5}$$

where $0 < \alpha \leq 1$. Then

$$1 - \alpha = 1 - \frac{1}{\tau_F / \Delta t + 1} = \frac{\tau_F}{\tau_F + \Delta t} \tag{C.6}$$

so that

$$y_n = \alpha x_n + (1 - \alpha) y_{n-1} \tag{C.7}$$

Equation (C.7) indicates that the filtered measurement is a weighted sum of the current measurement x_n and the filtered value at the previous sampling instant y_{n-1}. This operation is also called *single exponential smoothing*. Limiting cases for α are

$\alpha = 1$: No filtering (the filter output is the raw measurement x_n).

$\alpha = 0$: The measurement is ignored.

In the above limits, note that $\tau_F = \Delta t(1-\alpha)/\alpha$ by solving Equation (C.5); hence $\alpha = 1$ corresponds to a filter time constant of zero (no filtering).

Alternative expression for α in Equation (C.7) can be derived if the forward difference or other integration schemes for dy/dt are utilized.

C.2.2 Double Exponential Filter

Another popular digital filter is the double exponential or second-order filter, which offers some advantages for eliminating high frequency noise. The second-order filter is equivalent to two first-order filters in series where the second filter treats the output signal from the exponential filter in Equation (C.7). The second filter can be expressed as

$$\bar{y}_n = \gamma y_n + (1-\gamma)\bar{y}_{n-1} \tag{C.8}$$

$$\bar{y}_n = \gamma\alpha x_n + \gamma(1-\alpha)y_{n-1} + (1-\gamma)\bar{y}_{n-1} \tag{C.9}$$

Writing the filter equation in Equation (C.8) for the previous sampling instant gives

$$\bar{y}_{n-1} = \gamma y_{n-1} + (1-\gamma)\bar{y}_{n-2} \tag{C.10}$$

Solve for y_{n-1}:

$$y_{n-1} = \frac{1}{\gamma}\bar{y}_{n-1} - \frac{1-\gamma}{\gamma}\bar{y}_{n-2} \tag{C.11}$$

Substituting Equation (C.11) in to Equation (C.9) and rearranging gives the following expression for the double exponential filter:

$$\bar{y}_n = \gamma\alpha x_n + (2-\gamma-\alpha)\bar{y}_{n-1} - (1-\alpha)(1-\gamma)\bar{y}_{n-2} \tag{C.12}$$

A common simplification is to select $\gamma = \alpha$, yielding

$$\bar{y}_n = \alpha^2 x_n + 2(1-\alpha)\bar{y}_{n-1} - (1-\alpha)^2\bar{y}_{n-2} \tag{C.13}$$

The advantage of the double exponential filter over the exponential filter of Equation (C.7) is that it provides better filtering of high frequency noise, especially if $\gamma = \alpha$. A disadvantage of the double exponential filter is that it is more complicated than the exponential filter. Consequently, the single exponential filter has been more widely used in process control applications.

C.2.3 Moving Average Filter

A third type of digital filter is the moving-average filter, which averages a specified number of past data points, by giving equal weight to each data point. The moving-average filter is usually less effective than the exponential filter, which gives more weight to the most recent data.

The moving-average filter can be expressed as

$$y_n = \frac{1}{J} \sum_{i=n-J+1}^{n} x_i \qquad \text{(C.14)}$$

where J is the number of past data points that are being averaged. Equation (C.14) implies that the previous filtered value, y_{n-1}, can be expressed as

$$y_{n-1} = \frac{1}{J} \sum_{i=n-J}^{n-1} x_i \qquad \text{(C.15)}$$

Subtracting Equation (C.15) from Equation (C.14) gives the recursive form of the moving-average filter:

$$y_n = y_{n-1} + \frac{1}{J}(x_n - x_{n-1}) \qquad \text{(C.16)}$$

The exponential and moving-average filters are examples of low-pass filters, which are used to smooth noisy data by eliminating high-frequency noise.

C.2.4 Noise-Spike Filter

If a noisy measurement changes suddenly by a large amount and then returns to the original value (or close to it) at the next sampling instant, a *noise spike* is said to occur. In general, noise spikes can be caused by spurious electrical signals in the environment of the sensor. If noise spikes are not removed by filtering before the noisy measurement is sent to the controller, the controller will cause large, sudden changes in the manipulated variable.

Noise-spike filters (or *rate of change* filters) are used to limit how much the filtered output is permitted to change from one sampling instant to the next. If Δx denotes the maximum allowable change, the noise-spike filter can be written as

$$y_n = \begin{cases} x_n & \text{if } \left|x_n - y_{n-1}\right| \le \Delta x \\ y_{n-1} - \Delta x & \text{if } y_{n-1} - x_n > \Delta x \\ y_{n-1} + \Delta x & \text{if } y_{n-1} - x_n < -\Delta x \end{cases} \qquad \text{(C.17)}$$

If a large change in the measurement occurs, the filter replaces the measurement by the previous filter output plus (or minus) the maximum allowable change. This filter can also be used to detect instrument line, or an ADC "glitch."

Other type of more sophisticated digital filters are available but have not been commonly used in process control applications. These include high-pass filters and band-pass filters.

Appendix D

Tuning of Generic Model Controller

Lee and Sullivan (1988) outline a system for tuning GMC controllers based on choosing a target profile of the controlled variable, $x^*(t)$. This profile is characterized by two values, ξ and τ. Lee and Sullivan present a figure as shown in Figure D.1 that outlines the relative control performances of different combinations of ξ and τ. Having chosen the values of ξ and τ, the value of the two tuning constants, K_1 and K_2, are obtained using the following relationships:

$$K_1 = \frac{2\xi}{\tau}$$

$$K_2 = \frac{1}{\tau^2}$$

In tuning the GMC controller, because overshoot was undesirable, ξ was set to 10.0. The value of τ was obtained by examining the tuning charts given by Lee and Sullivan and recognizing that, with ξ=10.0, the controlled variable should cross the set point at approximately 0.25 τ.

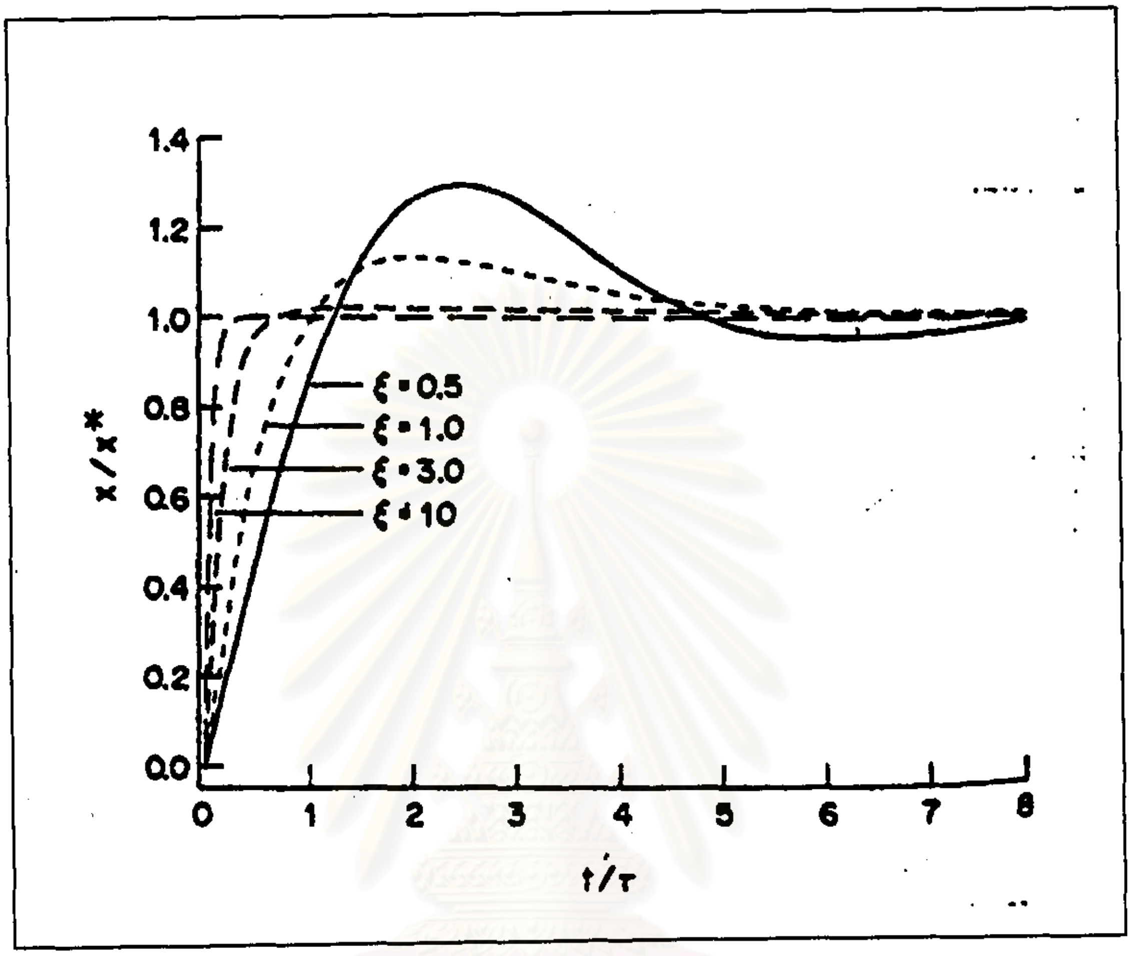

Figure D.1: Generalized GMC profile specificati

Vita

Jutatip Petcherdsak was born in Bangkok, Thailand, on September 16, 1973, the daughter of Vuttichai and Hong Petcherdsak. After completing her senior high school at Satri-Srisuriyothai School, Bangkok, Thailand, in 1992, she attended Chulalongkorn University, Bangkok, Thailand where she received the degree of Bachelor of Science in Chemical Technology on March 22, 1996. In June of 1996, she entered the Graduate School of Chulalongkorn University to pursue a Master of Engineering in Chemical Engineering, which she completed in October of 1999.