# APPENDICES

# Appendix A

## The estimation results of the Logistic function using rolling windows

Table A-1 The estimation results of the Logistic function using OLS and quadratic interpolation with the method of rolling window when the width of window is 3.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,801,249 | 0.2028 | 827,752,213 | 28.60 | 20.10 | 19.80 |
| 2 | 2,196,933 | -0.0336 | 5,608,479,871 | 130.40 | 22.02 | 21.72 |
| 3 | 2,694,513 | 0.0183 | 3,135,837,752 | 71.58 | 21.43 | 21.13 |
| 4 | 3,332,660 | 0.0185 | 2,686,089,049 | 70.01 | 21.28 | 20.98 |
| 5 | 4,103,375 | -0.0651 | 13,774,806,696 | 202.12 | 22.91 | 22.61 |
| 6 | 5,073,402 | -0.0279 | 7,186,289,977 | 92.02 | 22.26 | 21.96 |
| 7 | 6,264,092 | 0.0894 | 1,118,112,985 | 43.34 | 20.40 | 20.10 |
| 8 | 7,707,403 | 0.0011 | 4,522,025,036 | 59.72 | 21.80 | 21.50 |
| 9 | 9,496,986 | -0.0532 | 41,831,558,814 | 255.13 | 24.03 | 23.72 |
| 10 | 11,747,851 | 0.0224 | 876,007,444 | 38.01 | 20.16 | 19.86 |
| 11 | 14,465,776 | 0.0080 | 2,231,801,379 | 30.49 | 21.09 | 20.79 |
| 12 | 17,834,933 | 0.0106 | 4,601,992,414 | 40.01 | 21.82 | 21.52 |
| 13 | 22,001,390 | 0.0481 | 2,551,572,578 | 40.03 | 21.23 | 20.93 |
| 14 | 27,112,342 | -0.0205 | 60,623,759,035 | 244.80 | 24.40 | 24.10 |
| 15 | 33,446,352 | -0.0103 | 22,564,367,498 | 114.82 | 23.41 | 23.11 |
| 16 | 41,267,417 | -0.0139 | 18,288,384,911 | 61.21 | 23.20 | 22.90 |
| 17 | 50,901,314 | 0.0004 | 8,218,213,154 | 33.52 | 22.40 | 22.10 |
| 18 | 62,807,207 | 0.0391 | 3,530,561,467 | 44.85 | 21.55 | 21.25 |

Source: Calculation using Matlab version 2007.

Table A-2 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 4.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,479,900 | 0.1343 | 9.65E+08 | 30.23 | 19.80 | 19.65 |
| 2 | 1,814,234 | -0.006 | 6.27E+09 | 100.92 | 21.67 | 21.52 |
| 3 | 2,220,276 | 0.0117 | 4.3E+09 | 77.14 | 21.30 | 21.14 |
| 4 | 2,752,891 | -0.0277 | 7.1E+09 | 89.90 | 21.80 | 21.64 |
| 5 | 3,387,220 | -0.0432 | 1.47E+10 | 144.56 | 22.53 | 22.37 |
| 6 | 4,167,552 | 0.005 | 5.25E+09 | 62.05 | 21.49 | 21.34 |
| 7 | 5,138,693 | 0.0703 | 1.05E+09 | 45.71 | 19.88 | 19.73 |
| 8 | 6,329,590 | -0.0059 | 7.43E+09 | 51.73 | 21.84 | 21.69 |
| 9 | 7,800,831 | -0.0444 | 4.92E+10 | 217.54 | 23.73 | 23.58 |
| 10 | 9,627,524 | 0.0179 | 1.36E+09 | 36.86 | 20.14 | 19.99 |

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 11 | 11,854,787 | 0.0303 | 2.62E+09 | 31.08 | 20.80 | 20.65 |
| 12 | 14,625,119 | 0.0158 | 4.43E+09 | 36.13 | 21.33 | 21.17 |
| 13 | 18,035,085 | 0.0448 | 2.54E+09 | 42.65 | 20.77 | 20.62 |
| 14 | 22,220,133 | -0.025 | 9.77E+10 | 255.14 | 24.42 | 24.26 |
| 15 | 27,422,701 | -0.0209 | 4.7E+10 | 116.14 | 23.69 | 23.53 |
| 16 | 33,821,129 | -0.0022 | 1.39E+10 | 42.30 | 22.47 | 22.31 |
| 17 | 41,726,283 | 0.008 | 6.73E+09 | 29.86 | 21.74 | 21.59 |

Source: Calculation using Matlab version 2007.

Table A-3 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 5.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,481,169 | 0.1248 | 9.87E+08 | 34.00 | 19.50 | 19.42 |
| 2 | 1,853,969 | -0.0087 | 8.11E+09 | 108.31 | 21.61 | 21.53 |
| 3 | 2,240,391 | -0.0315 | 9.3E+09 | 98.12 | 21.74 | 21.67 |
| 4 | 2,786,427 | -0.0116 | 7.31E+09 | 67.57 | 21.50 | 21.43 |
| 5 | 3,414,637 | -0.0134 | 1.19E+10 | 96.52 | 21.99 | 21.92 |
| 6 | 4,206,172 | -0.0065 | 8.27E+09 | 57.35 | 21.63 | 21.55 |
| 7 | 5,185,882 | 0.0632 | 1.02E+09 | 49.73 | 19.54 | 19.46 |
| 8 | 6,390,211 | -0.0008 | 8.05E+09 | 43.84 | 21.60 | 21.52 |
| 9 | 7,870,546 | -0.0475 | 6.99E+10 | 201.90 | 23.76 | 23.68 |
| 10 | 9,697,855 | 0.0379 | 2.93E+09 | 40.32 | 20.59 | 20.51 |
| 11 | 11,962,451 | 0.0357 | 2.76E+09 | 34.17 | 20.53 | 20.45 |
| 12 | 14,750,244 | 0.0149 | 5.24E+09 | 33.57 | 21.17 | 21.09 |
| 13 | 18,188,491 | 0.0414 | 2.48E+09 | 45.43 | 20.42 | 20.34 |
| 14 | 22,418,866 | -0.035 | 1.75E+11 | 266.53 | 24.68 | 24.60 |
| 15 | 27,664,440 | -0.0119 | 4.11E+10 | 94.97 | 23.23 | 23.15 |
| 16 | 34,122,244 | 0.0042 | 1.19E+10 | 39.70 | 21.99 | 21.91 |

Source: Calculation using Matlab version 2007.

Table A-4 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 6.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,803,815 | 0.0980 | 1.11E+09 | 34.07 | 19.37 | 19.33 |
| 2 | 2,211,780 | -0.0486 | 1.53E+10 | 142.14 | 21.99 | 21.96 |
| 3 | 2,746,142 | -0.0158 | 9.66E+09 | 73.46 | 21.53 | 21.50 |
| 4 | 3,373,283 | 0.0137 | 5.84E+09 | 52.27 | 21.03 | 21.00 |
| 5 | 4,152,890 | -0.0230 | 1.76E+10 | 94.43 | 22.13 | 22.10 |
| 6 | 5,120,336 | -0.0082 | 1.05E+10 | 47.93 | 21.61 | 21.58 |
| 7 | 6,309,057 | 0.0655 | 1.25E+09 | 53.00 | 19.49 | 19.45 |
| 8 | 7,777,619 | -0.0059 | 1.15E+10 | 35.07 | 21.70 | 21.67 |
| 9 | 9,579,744 | -0.0278 | 5.43E+10 | 156.04 | 23.26 | 23.22 |
| 10 | 11,815,605 | 0.0435 | 3.47E+09 | 43.17 | 20.51 | 20.47 |
| 11 | 14,570,887 | 0.0352 | 2.77E+09 | 37.75 | 20.28 | 20.25 |
| 12 | 17,967,683 | 0.0134 | 6.35E+09 | 29.84 | 21.11 | 21.08 |
| 13 | 22,155,757 | 0.0318 | 2.61E+09 | 42.70 | 20.22 | 20.19 |
| 14 | 27,328,575 | -0.0278 | 1.71E+11 | 238.67 | 24.41 | 24.37 |
| 15 | 33,712,429 | -0.0065 | 3.87E+10 | 89.64 | 22.92 | 22.89 |

Source: Calculation using Matlab version 2007.

Table A-5 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 7.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,154,894 | 0.0420 | 2.06E+09 | 33.04 | 19.78 | 19.78 |
| 2 | 1,493,792 | -0.0312 | 1.53E+10 | 102.68 | 21.79 | 21.78 |
| 3 | 1,816,219 | 0.0096 | 7.67E+09 | 55.85 | 21.10 | 21.09 |
| 4 | 2,221,951 | 0.0023 | 8.13E+09 | 48.16 | 21.16 | 21.15 |
| 5 | 2,750,891 | -0.0238 | 2.12E+10 | 81.04 | 22.12 | 22.11 |
| 6 | 3,382,662 | -0.0015 | 1.03E+10 | 39.71 | 21.39 | 21.39 |
| 7 | 4,162,850 | 0.0582 | 1.2E+09 | 58.00 | 19.25 | 19.24 |
| 8 | 5,123,681 | 0.0103 | 8.8E+09 | 33.94 | 21.24 | 21.23 |
| 9 | 6,313,560 | -0.0201 | 5.24E+10 | 132.89 | 23.02 | 23.01 |
| 10 | 7,785,850 | 0.0437 | 3.61E+09 | 46.36 | 20.35 | 20.34 |
| 11 | 9,597,848 | 0.0339 | 2.76E+09 | 41.27 | 20.08 | 20.07 |
| 12 | 11,831,621 | 0.0058 | 1.08E+10 | 17.74 | 21.44 | 21.43 |
| 13 | 14,588,867 | 0.0369 | 4.23E+09 | 44.56 | 20.50 | 20.50 |
| 14 | 18,003,821 | -0.0230 | 1.71E+11 | 231.81 | 24.21 | 24.20 |

Source: Calculation using Matlab version 2007.

Table A-6 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 8.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,154,819 | 0.0477 | 2.01E+09 | 36.34 | 19.59 | 19.60 |
| 2 | 1,485,975 | -0.0040 | 1.28E+10 | 73.23 | 21.44 | 21.45 |
| 3 | 1,809,840 | -0.0014 | 1.04E+10 | 51.75 | 21.24 | 21.24 |
| 4 | 2,214,369 | -0.0005 | 9.78E+09 | 41.66 | 21.17 | 21.18 |
| 5 | 2,744,569 | -0.0164 | 2.15E+10 | 64.02 | 21.96 | 21.97 |
| 6 | 3,372,630 | -0.0046 | 1.27E+10 | 32.15 | 21.44 | 21.45 |
| 7 | 4,134,633 | 0.0709 | 5.19E+09 | 58.76 | 20.54 | 20.55 |
| 8 | 5,105,789 | 0.0159 | 8.12E+09 | 33.32 | 20.99 | 21.00 |
| 9 | 6,287,638 | -0.0174 | 5.69E+10 | 119.10 | 22.94 | 22.95 |
| 10 | 7,754,588 | 0.0426 | 3.68E+09 | 49.37 | 20.20 | 20.21 |
| 11 | 9,554,299 | 0.0265 | 3.33E+09 | 37.26 | 20.10 | 20.11 |
| 12 | 11,781,719 | 0.0115 | 9.34E+09 | 18.55 | 21.13 | 21.14 |
| 13 | 14,532,486 | 0.0398 | 5.89E+09 | 44.05 | 20.67 | 20.68 |

Source: Calculation using Matlab version 2007.

Table A-7 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 9.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,170,708 | 0.0657 | 2.13E+09 | 37.41 | 19.50 | 19.53 |
| 2 | 1,493,980 | -0.0137 | 1.67E+10 | 66.12 | 21.56 | 21.59 |
| 3 | 1,816,533 | -0.0038 | 1.23E+10 | 44.11 | 21.26 | 21.28 |
| 4 | 2,226,606 | 0.0050 | 9.69E+09 | 36.47 | 21.02 | 21.04 |
| 5 | 2,757,631 | -0.0185 | 2.61E+10 | 51.82 | 22.01 | 22.03 |
| 6 | 3,376,734 | 0.0114 | 1.05E+10 | 31.33 | 21.10 | 21.12 |
| 7 | 4,157,326 | 0.0736 | 6.75E+09 | 60.12 | 20.66 | 20.68 |
| 8 | 5,128,033 | 0.0169 | 8.26E+09 | 33.12 | 20.86 | 20.88 |
| 9 | 6,317,481 | -0.0160 | 6.28E+10 | 108.17 | 22.89 | 22.91 |
| 10 | 7,783,130 | 0.0357 | 3.71E+09 | 49.72 | 20.06 | 20.08 |
| 11 | 9,597,384 | 0.0314 | 4.47E+09 | 39.28 | 20.25 | 20.27 |
| 12 | 11,835,431 | 0.0153 | 8.58E+09 | 20.95 | 20.90 | 20.92 |

Source: Calculation using Matlab version 2007.

Table A-8 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 10.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,154,446 | 0.0486 | 2.5E+09 | 37.81 | 19.54 | 19.57 |
| 2 | 1,484,319 | -0.0150 | 1.94E+10 | 55.11 | 21.59 | 21.62 |
| 3 | 1,810,290 | 0.0019 | 1.22E+10 | 37.84 | 21.12 | 21.15 |
| 4 | 2,214,679 | 0.0014 | 1.16E+10 | 30.57 | 21.07 | 21.11 |
| 5 | 2,729,501 | -0.0020 | 2.08E+10 | 41.23 | 21.65 | 21.69 |
| 6 | 3,357,911 | 0.0175 | 9.79E+09 | 32.04 | 20.90 | 20.93 |
| 7 | 4,135,158 | 0.0721 | 7.44E+09 | 61.92 | 20.63 | 20.66 |
| 8 | 5,098,830 | 0.0168 | 8.61E+09 | 32.22 | 20.77 | 20.80 |
| 9 | 6,279,733 | -0.0203 | 8.14E+10 | 93.17 | 23.02 | 23.05 |
| 10 | 7,739,146 | 0.0403 | 5.79E+09 | 51.08 | 20.38 | 20.41 |
| 11 | 9,547,785 | 0.0345 | 5.75E+09 | 38.88 | 20.37 | 20.40 |

Source: Calculation using Matlab version 2007.

Table A-9 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 11.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,152,311 | 0.0414 | 2.73E+09 | 39.19 | 19.51 | 19.55 |
| 2 | 1,484,715 | -0.0083 | 1.94E+10 | 45.09 | 21.47 | 21.51 |
| 3 | 1,809,364 | -0.0014 | 1.44E+10 | 30.96 | 21.17 | 21.21 |
| 4 | 2,202,041 | 0.0165 | 1.02E+10 | 29.75 | 20.83 | 20.87 |
| 5 | 2,726,638 | 0.0048 | 1.93E+10 | 36.48 | 21.46 | 21.50 |
| 6 | 3,352,558 | 0.0190 | 9.72E+09 | 33.23 | 20.78 | 20.82 |
| 7 | 4,130,691 | 0.0696 | 7.95E+09 | 63.68 | 20.58 | 20.62 |
| 8 | 5,088,344 | 0.0112 | 1.13E+10 | 22.87 | 20.93 | 20.97 |
| 9 | 6,266,672 | -0.0136 | 7.34E+10 | 80.29 | 22.80 | 22.84 |
| 10 | 7,731,124 | 0.0432 | 7.89E+09 | 50.56 | 20.57 | 20.61 |

Source: Calculation using Matlab version 2007.

Table A-10 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 12.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,152,992 | 0.0432 | 2.7E+09 | 43.28 | 19.40 | 19.44 |
| 2 | 1,484,451 | -0.0109 | 2.25E+10 | 35.13 | 21.52 | 21.56 |
| 3 | 1,797,008 | 0.0139 | 1.25E+10 | 30.25 | 20.93 | 20.97 |
| 4 | 2,200,964 | 0.0221 | 9.67E+09 | 31.44 | 20.67 | 20.71 |
| 5 | 2,722,433 | 0.0071 | 1.93E+10 | 32.97 | 21.37 | 21.41 |
| 6 | 3,349,217 | 0.0193 | 9.84E+09 | 33.92 | 20.69 | 20.73 |
| 7 | 4,120,717 | 0.0619 | 7.64E+09 | 68.28 | 20.44 | 20.48 |
| 8 | 5,080,148 | 0.0164 | 1.05E+10 | 24.38 | 20.76 | 20.80 |
| 9 | 6,255,391 | -0.0087 | 6.82E+10 | 76.75 | 22.63 | 22.67 |

Source: Calculation using Matlab version 2007.

Table A-11 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 13.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,151,351 | 0.0365 | 2.99E+09 | 44.75 | 19.41 | 19.45 |
| 2 | 1,470,407 | 0.0052 | 1.9E+10 | 34.07 | 21.26 | 21.30 |
| 3 | 1,794,725 | 0.0197 | 1.18E+10 | 31.48 | 20.78 | 20.82 |
| 4 | 2,195,132 | 0.0234 | 9.59E+09 | 34.00 | 20.57 | 20.62 |
| 5 | 2,717,295 | 0.0080 | 1.98E+10 | 29.58 | 21.30 | 21.34 |
| 6 | 3,337,425 | 0.0142 | 1.18E+10 | 27.15 | 20.78 | 20.82 |
| 7 | 4,105,223 | 0.0648 | 1.13E+10 | 68.58 | 20.74 | 20.78 |
| 8 | 5,067,061 | 0.0200 | 1.03E+10 | 25.00 | 20.64 | 20.68 |

Source: Calculation using Matlab version 2007.

Table A-12 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 14.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,147,465 | 0.0492 | 5.26E+09 | 45.26 | 19.89 | 19.93 |
| 2 | 1,473,992 | 0.0116 | 1.78E+10 | 33.15 | 21.11 | 21.15 |
| 3 | 1,798,845 | 0.0211 | 1.17E+10 | 33.31 | 20.68 | 20.73 |
| 4 | 2,199,800 | 0.0235 | 9.64E+09 | 35.89 | 20.49 | 20.54 |
| 5 | 2,720,083 | 0.0036 | 2.37E+10 | 18.43 | 21.39 | 21.44 |
| 6 | 3,343,677 | 0.0195 | 1.15E+10 | 28.49 | 20.67 | 20.71 |
| 7 | 4,113,136 | 0.0664 | 1.49E+10 | 67.85 | 20.93 | 20.97 |

Source: Calculation using Matlab version 2007.

Table A-13 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 15.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,156,423 | 0.0527 | 6E+09 | 47.22 | 19.94 | 19.99 |
| 2 | 1,482,125 | 0.0136 | 1.77E+10 | 32.89 | 21.02 | 21.07 |
| 3 | 1,806,054 | 0.0213 | 1.18E+10 | 34.63 | 20.61 | 20.66 |
| 4 | 2,205,003 | 0.0182 | 1.11E+10 | 30.35 | 20.56 | 20.61 |
| 5 | 2,729,756 | 0.0093 | 2.17E+10 | 18.97 | 21.23 | 21.27 |
| 6 | 3,357,948 | 0.0231 | 1.16E+10 | 27.40 | 20.60 | 20.65 |

Source: Calculation using Matlab version 2007.

Table A-14 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 16.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,156,741 | 0.0520 | 6.25E+09 | 49.93 | 19.91 | 19.96 |
| 2 | 1,483,627 | 0.0142 | 1.79E+10 | 32.13 | 20.96 | 21.01 |
| 3 | 1,802,822 | 0.0162 | 1.35E+10 | 28.85 | 20.68 | 20.73 |
| 4 | 2,206,022 | 0.0232 | 1.13E+10 | 32.03 | 20.50 | 20.55 |
| 5 | 2,731,507 | 0.0134 | 2.05E+10 | 21.81 | 21.10 | 21.15 |

Source: Calculation using Matlab version 2007.

Table A-15 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 17.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,153,404 | 0.0503 | 6.4E+09 | 52.61 | 19.86 | 19.91 |
| 2 | 1,478,422 | 0.0095 | 2.06E+10 | 23.26 | 21.03 | 21.08 |
| 3 | 1,801,119 | 0.0213 | 1.34E+10 | 30.06 | 20.60 | 20.65 |
| 4 | 2,206,081 | 0.0267 | 1.18E+10 | 31.32 | 20.47 | 20.52 |

Source: Calculation using Matlab version 2007.

Table A-16 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 18.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,471,196 | 0.0431 | 6.38E+09 | 54.62 | 19.80 | 19.85 |
| 2 | 1,795,268 | 0.0149 | 1.99E+10 | 24.64 | 20.93 | 20.98 |
| 3 | 2,197,795 | 0.0247 | 1.38E+10 | 29.18 | 20.57 | 20.62 |

Source: Calculation using Matlab version 2007.

Table A-17 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 19.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,474,993 | 0.0468 | 8.8E+09 | 55.45 | 20.06 | 20.11 |
| 2 | 1,801,382 | 0.0188 | 1.94E+10 | 25.23 | 20.85 | 20.90 |

Source: Calculation using Matlab version 2007.

Table A-18 The estimation results of the Logistic Model using OLS and quadratic interpolation with the method of rolling window when the width of window is 20.

| Repeat | Mstar | beta | SSE | MAPE | AIC | BIC |
|---|---|---|---|---|---|---|
| 1 | 1,159,226 | 0.05 | 1.12E+10 | 54.90 | 20.24 | 20.29 |

Source: Calculation using Matlab version 2007.

# Appendix B

## Matlab code for OLS

```
% Analysis 1
% OLS vs EGLS
% Sub-analysis 1.1
% Fixed y-intercept
% Least Squares with Quadratic Interpolation

clear

%Please specify the numbers of data in use.
numbers_of_data=3


%Please speicy three upper bounds of the logistic function "M3".
M3= [200000 ; 500000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000

%Assume a constant variance "sigma"
sigma = 20000

%------------------------------------------------------------------


%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

load Sales

V=data(:,1)
T=data(:,2)

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3
safe = 99999

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
   end
```

```
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :)
grand_average=sum(monthly_average)/length(monthly_average)
seasonal_index=monthly_average*(100/grand_average)
s=seasonal_index
for r=2:1:year
   s = [s ; seasonal_index]
end
k=length(V)
s=s(1:k,:)
deseasonalized_V=V*100./(s)
V2=deseasonalized_V

%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000

%Calculate A to fix y=V0 at T=0

V=V2

V=V(1:numbers_of_data , :)
T=T(1:numbers_of_data , :)

A=(M/V(1,:))-1

V/M
1-V/M
(V/M)./(1-(V/M))

Y=log((V/M)./(1-(V/M)))-log(1/A)

%Step 4: Ordinary Least Squares
%Assume alpha = zero to fix y-intercept at Vo when T=0

beta=(inv(T'*T))*(T'*Y)


%Step 5: Forecast

exp(-beta*T)
A
A.*exp(-beta*T)
```

```
1+(A.*exp(-beta*T))
F=M./(1+(A.*exp(-beta*T)))


%Step 6: Sum squared error (SSE)

Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)

%Step 7: Replace L with SSE
L=SSE

%Step 8: Collect records

results= [ M L ]
LL1=results(:,2)


%-----end of section 1-----------------------------------

%Section 2: Repeat with M3

for k=1:1:length(M3)
k
M=M3(k,:)
A=(M/V(1,:))-1
Y=log((V/M)./(1-(V/M)))-log(1/A)
beta=(inv(T'*T))*(T'*Y)
F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L ]
end

% Collect records

results=results(2:end, :)
LL2= results(:,2)

%-----end of section 2--------------------------------------

%Section 3: Searching algorithm
%Now we are using Quadratic Interpolation
%to search for optimum "M" or the upper bound of the S-curve
```

```
%x is the upper bounds of logistic function (S-curve)
x1=results(1,1)
x2=results(2,1)
x3=results(3,1)

% y is the value of likelihood function.
y1=results(1,2)
y2=results(2,2)
y3=results(3,2)

omega=(x1-x2)*(x2-x3)*(x3-x1)

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
   disp('The function is not quadratic, but linear instead.')
end
if a>0
   disp('The function has no maxima, but the minima instead.')
end

%----end of section 3-------------------------------------------

%Section 4: Automatic searching
%This section will use the iteration methods to run an automatic searching
%until we find optimal "M".



Mstar=-b/(2*a)

Mpast=Mstar %To send the value for comparison to Mstar of next round, not this
round.

%Step1:  Find likelihood value of Mstar

M=Mstar
A=(M/V(1,:))-1
Y=log((V/M)./(1-(V/M)))-log(1/A)
beta=(inv(T'*T))*(T'*Y)
F=M./(1+(A.*exp(-beta*T)))
```

```
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L ]
L2=L
Lpast=L2

%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.

diffM=M3-Mstar

diff1=min(diffM)

dd=999


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ]


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end



%Case2: Mstar is at extreme of M3
%And Mstar is smaller than other members.
if Mstar<min(M3)
   disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ]


get=[ 999 999 ]
for r=1:1:length(results)
```

```
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end



%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
   disp('Mstar is at the middle and nearest to the top')
   MM= [ M3(2,:) ; Mstar ; M3(3,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end



%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
   disp('Mstar is at the middle and nearest to the bottom')
   MM= [ M3(1,:) ; Mstar ; M3(2,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
```

```
   end

   end
end
end


get=get(2:end ,:)

% Step3: Sorting matrix M by ascending order

for k=1:1:length(get)
   if get(k,1)==min(get(:,1))
      get1=get(k,:)

   elseif get(k,1)==max(get(:,1))
      get3=get(k,:)

   else
      get2=get(k,:)
   end

end

get = [ get1 ; get2 ; get3 ]
LL3=get(:,2)


%Step3: Interpolation again

results=get

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1)
x2=results(2,1)
x3=results(3,1)

% y is the value of likelihood function.
y1=results(1,2)
y2=results(2,2)
y3=results(3,2)

omega=(x1-x2)*(x2-x3)*(x3-x1)

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega
```

```
if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
   disp('The function is not quadratic, but linear instead.')
end
if a>0
   disp('The function has no maxima, but the minima instead.')
end

Mstar=-b/(2*a)
LL=LL3



%---------------end of section 4-----------------


%Section 5: Iteration

for iteration=3:1:maxiteration
%Monitoring the process
Mstar
Mpast
M3=get(:,1)
LL
L
Lpast

disp('You are using Mazimum Likelihood estimation for the S-curve')

%Step1:  Find likelihood value of Mstar
M=Mstar   ;
A=(M/V(1,:))-1 ;
Y=log((V/M)./(1-(V/M))-log(1/A);
beta=(inv(T'*T))*(T'*Y);
F=M./(1+(A.*exp(-beta*T)));
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;
results= [ results ; M L ];

%Monitoring stopping criteria
L
Lpast
```

```
increment=L-Lpast
iteration



%%----------------------------------------------------------
%Stopping criteria

%Please specify the stopping criteria.
stop= 0.001*Lpast ; %Difference between L and Lpast
if abs(increment)>stop
%%---------------------------------------------------------
Lpast=L



%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.

diffM=M3-Mstar;

diff1=min(diffM);

dd=999;


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ];


get=[ 999 999 ];
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case2: Mstar is at extreme of M3
```

```
%And Mstar is smaller than other members.
if Mstar<min(M3)
   disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ];


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
   disp('Mstar is at the middle and nearest to the top')
   MM= [ M3(2,:) ; Mstar ; M3(3,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
```

```
    disp('Mstar is at the middle and nearest to the bottom')
    MM= [ M3(1,:) ; Mstar ; M3(2,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
    for k=1:1:length(MM)
    if MM(k,:)==results(r,1)
        get=[ get ; results(r,:)];
    end

    end
end
end


get=get(2:end ,:)



% Step3: Sorting matrix M by ascending order

for k=1:1:length(get)
    if get(k,1)==min(get(:,1))
        get1=get(k,:);

    elseif get(k,1)==max(get(:,1))
        get3=get(k,:);

    else
        get2=get(k,:);
    end

end

get = [ get1 ; get2 ; get3 ];
LL=get(:,2)


%Step3: Interpolation again

results=get

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1);
x2=results(2,1);
x3=results(3,1);
```

```
% y is the value of likelihood function.
y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

omega=(x1-x2)*(x2-x3)*(x3-x1);

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega;
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega;
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega;

Mpast=Mstar;

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a);
end
if a==0
disp('The function is not quadratic, but linear instead.')
end
if a>0
disp('The function has no maxima, but the minima instead.')
end
Mstar=-b/(2*a);



end
end

%---------------end of section 5-----------------




%Summary

disp('---------------------------------------------------------------------------')
disp('Summary report')
disp('Congratulations! You have finished Maximum Likelihood estimation of the S-Curve')


iteration
Mstar
beta
```

```
if increment<stop
disp('The model reaches the extrema, then the stopping criteria works.')
end

if a==0
disp('Warning: The function is not quadratic, but linear instead.')
end
if a>0
disp('Warning: The function has no maxima, but the minima instead.')
end
if a<0
disp('The function has the maxima.')
end


disp('Thank you very much for using KS Software and KSAN approach')
disp('-------------------------------------------------------------------------------')
disp('-------------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')


%----------------------------
%Calculate SSE (within model)

V=V2(:,1)
V=V(1:numbers_of_data,:)
T=data(:,2)
T=T(1:numbers_of_data,:)

Vhat=Mstar./(1+(A./(1+exp(-beta*T))))
SSE=sum((V-Vhat).^2)

N=length(V)
k=1

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)


%Calculate MAPE for the out-of-sample-test

V=V2(:,1)
T=data(:,2)
N=length(V)

Vhat=Mstar./(1+(A./(1+exp(-beta*T))))
```

```
number_out_sample=32-numbers_of_data

if number_out_sample>0

out_sample=V(32+1-number_out_sample:end,:)

Error=out_sample-Vhat(32+1-number_out_sample:end,:)
APE=abs(Error./out_sample)*100
N=length(out_sample)
MAPE=sum(APE)./N

print=[ numbers_of_data Mstar beta  SSE MAPE  AIC  BIC ]

else

print=[ numbers_of_data Mstar beta SSE 0  AIC  BIC]

end
```

# Appendix C

## Matlab code for EGLS

```
% Analysis 1
% OLS vs EGLS
% Sub-analysis 1.2
% Least Squares with Quadratic Interpolation

clear

%Please specify the numbers of data in use.
numbers_of_data=3


%Please speicy three upper bounds of the logistic function "M3".
M3= [200000 ; 500000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000

%Assume a constant variance "sigma"
sigma = 20000

%----------------------------------------------------------------


%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

load Sales

V=data(:,1)
T=data(:,2)

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3
safe = 99999

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
   end
```

```
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :)
grand_average=sum(monthly_average)/length(monthly_average)
seasonal_index=monthly_average*(100/grand_average)
s=seasonal_index
for r=2:1:year
   s = [s ; seasonal_index]
end
k=length(V)
s=s(1:k,:)
deseasonalized_V=V*100./(s)
V2=deseasonalized_V

%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000

%Calculate A to fix y=V0 at T=0

V=V2

V=V(1:numbers_of_data , :)
T=T(1:numbers_of_data , :)

A=(M/V(1,:))-1

%Step 4: EGLS
%Assume alpha = zero to fix y-intercept at Vo when T=0

P1=V./M ;
P2=(1-P1);
v=log(P1./P2)-log(1/A);
N=numbers_of_data;
omega=1./(N.*(P1*P2'));
%omega=diag(omega);
%omega=diag(omega);
a=inv(T'*inv(omega)*T);
b=(T'*inv(omega)*v);
beta_EGLS_start=a*b;

beta_EGLSnew=beta_EGLS_start;
beta_EGLSold=9.999;
beta_EGLS=beta_EGLSnew;
```

```
beta=beta_EGLS

%Step 5: Forecast

exp(-beta*T)
A
A.*exp(-beta*T)
1+(A.*exp(-beta*T))
F=M./(1+(A.*exp(-beta*T)))


%Step 6: Sum squared error (SSE)

Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)

%Step 7: Replace L with SSE
L=SSE

%Step 8: Collect records

results= [ M L ]
LL1=results(:,2)


%-----end of section 1-----------------------------------

%Section 2: Repeat with M3

for k=1:1:length(M3)
k
M=M3(k,:)
A=(M/V(1,:))-1

P1=V./M ;
P2=(1-P1);
v=log(P1./P2)-log(1/A);
N=numbers_of_data;
omega=1./(N.*(P1*P2'));
%omega=diag(omega);
%omega=diag(omega);
a=inv(T'*inv(omega)*T);
b=(T'*inv(omega)*v);
beta_EGLS_start=a*b;

beta_EGLSnew=beta_EGLS_start;
```

```
beta_EGLSold=9.999;
beta_EGLS=beta_EGLSnew;
beta=beta_EGLS


F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L ]
end

% Collect records

results=results(2:end, :)
LL2= results(:,2)

%-----end of section 2--------------------------------------

%Section 3: Searching algorithm
%Now we are using Quadratic Interpolation
%to search for optimum "M" or the upper bound of the S-curve

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1)
x2=results(2,1)
x3=results(3,1)

% y is the value of likelihood function.
y1=results(1,2)
y2=results(2,2)
y3=results(3,2)

omega=(x1-x2)*(x2-x3)*(x3-x1)

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
   disp('The function is not quadratic, but linear instead.')
end
if a>0
```

```
   disp('The function has no maxima, but the minima instead.')
end

%----end of section 3-----------------------------------------

%Section 4: Automatic searching
%This section will use the iteration methods to run an automatic searching
%until we find optimal "M".



Mstar=-b/(2*a)

Mpast=Mstar %To send the value for comparison to Mstar of next round, not this
round.

%Step1:  Find likelihood value of Mstar

M=Mstar
A=(M/V(1,:))-1

P1=V./M ;
P2=(1-P1);
v=log(P1./P2)-log(1/A);
N=numbers_of_data;
omega=1./(N.*(P1*P2'));
%omega=diag(omega);
%omega=diag(omega);
a=inv(T'*inv(omega)*T);
b=(T'*inv(omega)*v);
beta_EGLS_start=a*b;

beta_EGLSnew=beta_EGLS_start;
beta_EGLSold=9.999;
beta_EGLS=beta_EGLSnew;
beta=beta_EGLS


F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L ]
L2=L
Lpast=L2
```

```
%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.

diffM=M3-Mstar

diff1=min(diffM)

dd=999


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ]


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end



%Case2: Mstar is at extreme of M3
%And Mstar is smaller than other members.
if Mstar<min(M3)
   disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ]


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end
```

```
%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
   disp('Mstar is at the middle and nearest to the top')
   MM= [ M3(2,:) ; Mstar ; M3(3,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end



%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
   disp('Mstar is at the middle and nearest to the bottom')
   MM= [ M3(1,:) ; Mstar ; M3(2,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)]
   end

   end
end
end


get=get(2:end ,:)
```

```
% Step3: Sorting matrix M by ascending order

for k=1:1:length(get)
    if get(k,1)==min(get(:,1))
        get1=get(k,:)

    elseif get(k,1)==max(get(:,1))
        get3=get(k,:)

    else
        get2=get(k,:)
    end

end

get = [ get1 ; get2 ; get3 ]
LL3=get(:,2)


%Step3: Interpolation again

results=get

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1)
x2=results(2,1)
x3=results(3,1)

% y is the value of likelihood function.
y1=results(1,2)
y2=results(2,2)
y3=results(3,2)

omega=(x1-x2)*(x2-x3)*(x3-x1)

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
    disp('The function is not quadratic, but linear instead.')
end
if a>0
```

```
    disp('The function has no maxima, but the minima instead.')
end

Mstar=-b/(2*a)
LL=LL3



%---------------end of section 4-----------------


%Section 5: Iteration

for iteration=3:1:maxiteration
%Monitoring the process
Mstar
Mpast
M3=get(:,1)
LL
L
Lpast

disp('You are using Mazimum Likelihood estimation for the S-curve')

%Step1:  Find likelihood value of Mstar
M=Mstar   ;
A=(M/V(1,:))-1 ;

P1=V./M ;
P2=(1-P1);
v=log(P1./P2)-log(1/A);
N=numbers_of_data;
omega=1./(N.*(P1*P2'));
%omega=diag(omega);
%omega=diag(omega);
a=inv(T'*inv(omega)*T);
b=(T'*inv(omega)*v);
beta_EGLS_start=a*b;

beta_EGLSnew=beta_EGLS_start;
beta_EGLSold=9.999;
beta_EGLS=beta_EGLSnew;
beta=beta_EGLS



F=M./(1+(A.*exp(-beta*T)));
```

```
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;
results= [ results ; M L ];

%Monitoring stopping criteria
L
Lpast
increment=L-Lpast
iteration



%%------------------------------------------------------------
%Stopping criteria

%Please specify the stopping criteria.
stop= 0.001*Lpast ; %Difference between L and Lpast
if abs(increment)>stop
%%-----------------------------------------------------------
Lpast=L



%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.

diffM=M3-Mstar;

diff1=min(diffM);

dd=999;


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ];


get=[ 999 999 ];
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
```

```
    end

    end
end
end



%Case2: Mstar is at extreme of M3
%And Mstar is smaller than other members.
if Mstar<min(M3)
    disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ];


get=[ 999 999 ]
for r=1:1:length(results)
    for k=1:1:length(MM)
    if MM(k,:)==results(r,1)
        get=[ get ; results(r,:)];
    end

    end
end
end



%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
    disp('Mstar is at the middle and nearest to the top')
    MM= [ M3(2,:) ; Mstar ; M3(3,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
    for k=1:1:length(MM)
    if MM(k,:)==results(r,1)
        get=[ get ; results(r,:)];
    end

    end
end
end
```

```
%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
   disp('Mstar is at the middle and nearest to the bottom')
   MM= [ M3(1,:) ; Mstar ; M3(2,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end


get=get(2:end ,:)



% Step3: Sorting matrix M by ascending order

for k=1:1:length(get)
   if get(k,1)==min(get(:,1))
      get1=get(k,:);

   elseif get(k,1)==max(get(:,1))
      get3=get(k,:);

   else
      get2=get(k,:);
   end

end

get = [ get1 ; get2 ; get3 ];
LL=get(:,2)
```

```
%Step3: Interpolation again

results=get

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

% y is the value of likelihood function.
y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

omega=(x1-x2)*(x2-x3)*(x3-x1);

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega;
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega;
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega;

Mpast=Mstar;

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a);
end
if a==0
disp('The function is not quadratic, but linear instead.')
end
if a>0
disp('The function has no maxima, but the minima instead.')
end
Mstar=-b/(2*a);


end
end

%---------------end of section 5-----------------



%Summary

disp('--------------------------------------------------------------------------------')
```

```
disp('Summary report')
disp('Congratulations! You have finished Maximum Likelihood estimation of the S-Curve')


iteration
Mstar
beta

if increment<stop
disp('The model reaches the extrema, then the stopping criteria works.')
end

if a==0
disp('Warning: The function is not quadratic, but linear instead.')
end
if a>0
disp('Warning: The function has no maxima, but the minima instead.')
end
if a<0
disp('The function has the maxima.')
end


disp('Thank you very much for using KS Software and KSAN approach')
disp('------------------------------------------------------------------------------')
disp('------------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')


%----------------------------
%Calculate SSE (within model)

V=V2(:,1)
V=V(1:numbers_of_data,:)
T=data(:,2)
T=T(1:numbers_of_data,:)

Vhat=Mstar./(1+(A./(1+exp(-beta*T))))
SSE=sum((V-Vhat).^2)

N=length(V)
k=1

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)
```

```
%Calculate MAPE for the out-of-sample-test

V=V2(:,1)
T=data(:,2)
N=length(V)

Vhat=Mstar./(1+(A./(1+exp(-beta*T))))

number_out_sample=32-numbers_of_data

if number_out_sample>0

out_sample=V(32+1-number_out_sample:end,:)

Error=out_sample-Vhat(32+1-number_out_sample:end,:)
APE=abs(Error./out_sample)*100
N=length(out_sample)
MAPE=sum(APE)./N

print=[ numbers_of_data Mstar beta  SSE MAPE  AIC  BIC ]

else

print=[ numbers_of_data Mstar beta SSE 0  AIC  BIC]

end
```

# Appendix D

## Matlab code for Rolling Windows

```
% Analysis 5
% Cumulative observations vs Rolling windows
% Sub-analysis 5.1
% Logistic model
% Least squares with Quadratic Interpolation


%The design is like this:
%We will choose the width of the rolling window first.
%Then we will begin the window from observation 1.
%And forecast for next 12 observations (12 months--- 1 year).
%This is counted as 1 repeat.
%The number of repeats are not equal when the width of windows are different.
%This is because we have only 32 observations in total.
%The maximum width of window is 20.But we will have only 1 repeat for it.
%To ensure that we have more repeats for further statistical comparisons,
%We will stop the width of window at 15.
%The mimimum width of window is 3.
% This is the min no of obs required by regular process of non-linear analysis.
%Lets' begin.


clear

%Please specify the numbers of data in use.

width_of_window=5  %From 3 to 15
numbers_of_data=width_of_window

number_of_forecast=12  %Fixed at 12

number_of_repeat=32-numbers_of_data-number_of_forecast


%Please speicy three upper bounds of the logistic function "M3".
M3= [200000 ; 500000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000

%Assume a constant variance "sigma"
sigma = 20000

%---------------------------------------------------------------
```

```
%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

repeat=1

%for repeat=1:1:(32-width_of_window-number_of_forecast)

print=[ 9999 9999 9999  9999 9999  9999  9999 ];

load Sales

V=data(:,1);
T=data(:,2);

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3;
safe = 99999;

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
   end
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :);
grand_average=sum(monthly_average)/length(monthly_average);
seasonal_index=monthly_average*(100/grand_average);
s=seasonal_index;
for r=2:1:year
   s = [s ; seasonal_index];
end
k=length(V);
s=s(1:k,:);
deseasonalized_V=V*100./(s);
V2=deseasonalized_V;


maxrepeat=32-width_of_window-number_of_forecast+1
%maxrepeat=1
```

```
for repeat=1:1:maxrepeat



%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000;

%Starting points

V=V2;

V=V(repeat:numbers_of_data+(repeat-1) , :)
T=T(repeat:numbers_of_data+(repeat-1) , :)

A=(M/V(1,:))-1;

V/M;
1-V/M;
(V/M)./(1-(V/M));

Y=log((V/M)./(1-(V/M)))-log(1/A);

%Step 4: Ordinary Least Squares
%Assume alpha = zero to fix y-intercept at Vo when T=0

beta=(inv(T'*T))*(T'*Y);


%Step 5: Forecast

exp(-beta*T);
A;
A.*exp(-beta*T);
1+(A.*exp(-beta*T));
F=M./(1+(A.*exp(-beta*T)));


%Step 6: Sum squared error (SSE)

Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);

%Step 7: Replace L with SSE
```

```
L=SSE;

%Step 8: Collect records

results= [ M L ];
LL1=results(:,2);


%-----end of section 1-----------------------------------

%Section 2: Repeat with M3

for k=1:1:length(M3)
k;
M=M3(k,:)  ;
A=(M/V(1,:))-1;
Y=log((V/M)./(1-(V/M)))-log(1/A);
beta=(inv(T'*T))*(T'*Y);
F=M./(1+(A.*exp(-beta*T)));
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;
results= [ results ; M L ];
end

% Collect records

results=results(2:end, :);
LL2= results(:,2);

%-----end of section 2--------------------------------------

%Section 3: Searching algorithm
%Now we are using Quadratic Interpolation
%to search for optimum "M" or the upper bound of the S-curve

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

% y is the value of likelihood function.
y1=results(1,2);
y2=results(2,2);
y3=results(3,2);
```

```
omega=(x1-x2)*(x2-x3)*(x3-x1);

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega;
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega;
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega;

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
   disp('The function is not quadratic, but linear instead.')
end
if a>0
   disp('The function has no maxima, but the minima instead.')
end

%----end of section 3------------------------------------------

%Section 4: Automatic searching
%This section will use the iteration methods to run an automatic searching
%until we find optimal "M".



Mstar=-b/(2*a);

Mpast=Mstar ;%To send the value for comparison to Mstar of next round, not this
round.

%Step1:  Find likelihood value of Mstar

M=Mstar   ;
A=(M/V(1,:))-1;
Y=log((V/M)./(1-(V/M)))-log(1/A);
beta=(inv(T'*T))*(T'*Y);
F=M./(1+(A.*exp(-beta*T)));
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;
results= [ results ; M L ];
L2=L;
Lpast=L2;

%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.
```

```
diffM=M3-Mstar;

diff1=min(diffM);

dd=999;


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ]


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case2: Mstar is at extreme of M3
%And Mstar is smaller than other members.
if Mstar<min(M3)
   disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ];


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end
```

```
%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
   disp('Mstar is at the middle and nearest to the top')
   MM= [ M3(2,:) ; Mstar ; M3(3,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
   disp('Mstar is at the middle and nearest to the bottom')
   MM= [ M3(1,:) ; Mstar ; M3(2,:) ]

get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end


get=get(2:end ,:)

% Step3: Sorting matrix M by ascending order
```

```
for k=1:1:length(get)
    if get(k,1)==min(get(:,1))
        get1=get(k,:)

    elseif get(k,1)==max(get(:,1))
        get3=get(k,:);

    else
        get2=get(k,:);
    end

end

get = [ get1 ; get2 ; get3 ];
LL3=get(:,2);


%Step3: Interpolation again

results=get;

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

% y is the value of likelihood function.
y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

omega=(x1-x2)*(x2-x3)*(x3-x1);

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega;
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega;
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega;

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a)
end
if a==0
    disp('The function is not quadratic, but linear instead.')
end
if a>0
    disp('The function has no maxima, but the minima instead.')
end
```

```
Mstar=-b/(2*a)
LL=LL3



%---------------end of section 4----------------


%Section 5: Iteration

for iteration=3:1:maxiteration
%Monitoring the process
Mstar;
Mpast;
M3=get(:,1);
LL;
L;
Lpast;

disp('You are rolling the S-curve windows.')
repeat

%Step1:  Find likelihood value of Mstar
M=Mstar   ;
A=(M/V(1,:))-1 ;
Y=log((V/M)./(1-(V/M)))-log(1/A);
beta=(inv(T'*T))*(T'*Y);
F=M./(1+(A.*exp(-beta*T)));
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;
results= [ results ; M L ];

%Monitoring stopping criteria
L;
Lpast;
increment=L-Lpast;
iteration;


%%--------------------------------------------------------
%Stopping criteria

%Please specify the stopping criteria.
stop= 0.001*Lpast ; %Difference between L and Lpast
```

```
if abs(increment)>stop
%%-------------------------------------------------------
Lpast=L;



%Step2: Inserting Mstar into matrix M
%Find the nearest points nearby Mstar.

diffM=M3-Mstar;

diff1=min(diffM);

dd=999;


%Case1: Mstar is at extreme of M3
%And Mstar is larger than other members.
if Mstar>max(M3)
   disp('Mstar is at the top.')
MM=[  M3(2,:) ; M3(3,:) ; Mstar ; ];


get=[ 999 999 ];
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
      get=[ get ; results(r,:)];
   end

   end
end
end



%Case2: Mstar is at extreme of M3
%And Mstar is smaller than other members.
if Mstar<min(M3)
   disp('Mstar is at the bottom.')
MM = [  Mstar ; M3(1,:) ; M3(2,:)  ];


get=[ 999 999 ]
for r=1:1:length(results)
   for k=1:1:length(MM)
   if MM(k,:)==results(r,1)
```

```
        get=[ get ; results(r,:)];
    end

    end
end
end



%Case3: Mstar is somewhere at the middle of M3
%And it is located nearest to the top
%One positive sign and two negative signs in diffM
%Eliminate the bottom.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)>0
    disp('Mstar is at the middle and nearest to the top')
    MM= [ M3(2,:) ; Mstar ; M3(3,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
    for k=1:1:length(MM)
    if MM(k,:)==results(r,1)
        get=[ get ; results(r,:)];
    end

    end
end
end



%Case4: Mstar is somewhere at the middle of M3
%And it is located nearest to the bottom
%Two positive signs and one negative sign in diffM
%Eliminate the top.

if Mstar<max(M3)& Mstar>min(M3)& prod(diffM)<0
    disp('Mstar is at the middle and nearest to the bottom')
    MM= [ M3(1,:) ; Mstar ; M3(2,:) ];

get=[ 999 999 ];
for r=1:1:length(results)
    for k=1:1:length(MM)
    if MM(k,:)==results(r,1)
        get=[ get ; results(r,:)];
    end
```

```
   end
end
end


get=get(2:end ,:)



% Step3: Sorting matrix M by ascending order

for k=1:1:length(get)
   if get(k,1)==min(get(:,1))
      get1=get(k,:);

   elseif get(k,1)==max(get(:,1))
      get3=get(k,:);

   else
      get2=get(k,:);
   end

end

get = [ get1 ; get2 ; get3 ];
LL=get(:,2);


%Step3: Interpolation again

results=get;

%x is the upper bounds of logistic function (S-curve)
x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

% y is the value of likelihood function.
y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

omega=(x1-x2)*(x2-x3)*(x3-x1);

a= ( (x3-x2)*y1 + (x1-x3)*y2 + (x2-x1)*y3 )/omega;
b= ( (x2^2 - x3^2)*y1 +  (x3^2 - x1^2)*y2 + (x1^2 - x2^2)*y3)/omega;
c= ( x2*x3*(x3-x2)*y1 + x3*x1*(x1-x3)*y2 + x1*x2*(x2-x1)*y3 )/omega;
```

```
Mpast=Mstar;

if a<0 %to ensure that the function has the maxima.
Mstar=-b/(2*a);
end
if a==0
disp('The function is not quadratic, but linear instead.')
end
if a>0
disp('The function has no maxima, but the minima instead.')
end
Mstar=-b/(2*a);



end
end

%---------------end of section 5-----------------




%Summary

disp('-------------------------------------------------------------------------------')
disp('Summary report')
disp('Congratulations! You have finished Maximum Likelihood estimation of the S-
Curve')


iteration;
Mstar;
beta;

if increment<stop
disp('The model reaches the extrema, then the stopping criteria works.')
end

if a==0
disp('Warning: The function is not quadratic, but linear instead.')
end
if a>0
disp('Warning: The function has no maxima, but the minima instead.')
end
```

```
if a<0
disp('The function has the maxima.')
end


disp('Thank you very much for using KS Software and KSAN approach')
disp('-----------------------------------------------------------------------------')
disp('-----------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')


%----------------------------
%Calculate SSE (within model)

V=V2(:,1);
V=V(repeat:numbers_of_data+(repeat-1),:);
T=data(:,2);
T=T(repeat:numbers_of_data+(repeat-1),:);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));
SSE=sum((V-Vhat).^2);

N=length(V);
k=1;

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)


%Calculate MAPE for the out-of-sample-test

V=V2(:,1);
T=data(:,2);
N=length(V);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));

number_out_sample=number_of_forecast;

if number_out_sample>0

out_sample=V(width_of_window+repeat:width_of_window+repeat+12-1,:);

Error=out_sample-Vhat(width_of_window+repeat:width_of_window+repeat+12-1,:);
APE=abs(Error./out_sample)*100;
N=length(out_sample);
```

```
MAPE=sum(APE)./N

print=[ print ; repeat Mstar beta  SSE MAPE  AIC  BIC ]

else

print=[ print ; repeat Mstar beta SSE 0  AIC  BIC]

end

end

repeat
print=print(2:end,:)
length(print)
width_of_window
length(out_sample)
out_sample(end,:)
```

# Appendix E

## Matlab code for Quasi-Newton

```
% Analysis 6
% Quasi-Newton vs Gauss-Newton
% Sub-analysis 6.1
% Quasi-Newton
% Logistic model
% Least squares

clear

%Please specify the numbers of data in use.
numbers_of_data=32


%Please speicy three upper bounds of the logistic function "M3".
M3= [800000 ; 900000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000



%----------------------------------------------------------------


%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

load Sales

V=data(:,1)
T=data(:,2)

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3
safe = 99999

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
```

```
   end
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :)
grand_average=sum(monthly_average)/length(monthly_average)
seasonal_index=monthly_average*(100/grand_average)
s=seasonal_index
for r=2:1:year
   s = [s ; seasonal_index]
end
k=length(V)
s=s(1:k,:)
deseasonalized_V=V*100./(s)
V2=deseasonalized_V

%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000

%Calculate A to fix y=V0 at T=0

V=V2

V=V(1:numbers_of_data , :)
T=T(1:numbers_of_data , :)

A=(M/V(1,:))-1



% Now we are using Quasi-Newton to estimate both M and beta at the same time.
% We will fix y-intercept, therefore A is prior calculated.
% We will use Least Squares to be objective function.



%To find SSE and beta for the related M, we will use the OLS method.
%It is the efficient way to find beta from M, otherwise we guess beta from nothing.

results=[ 999 999 999]

for iteration=1:1:length(M3)
k=iteration
M=M3(k,:)
```

```
A=(M/V(1,:))-1
Y=log((V/M)./(1-(V/M)))-log(1/A)
beta=(inv(T'*T))*(T'*Y)
F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L beta ]
end

results=results(2:end,:)


% We extract all elements from the results.

x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

z1=results(1,3);
z2=results(2,3);
z3=results(3,3);


%Quasi-Newton

H=eye(2)


% Find slopes
% Notes: w in the text means parameters to be estimated which are M and beta

% Slope of M
delta_M1=x2-x3
slope_M1=(y2-y3)/delta_M1

delta_M2=x1-x2
slope_M2=(y1-y2)/delta_M2

% Slope of beta
delta_beta1=z2-z3
slope_beta1=(y2-y3)/delta_beta1
```

```
delta_beta2=z1-z2
slope_beta2=(y1-y2)/delta_beta2

%Calculate vi and ui

vi= [x2-x3 ; z2-z3 ]
ui= [slope_M2-slope_M1 ; slope_beta2-slope_beta1 ]

%Calculate the next H

H=H + ((vi*vi')/(vi'*ui))-((H*ui*ui'*H)/(ui'*H*ui))

%Calculate d

g= [slope_M1 ; slope_beta1]
d= (-H)*g

%Calculate next M and beta

parameter= [ x3 ; z3 ]
parameter=parameter+d

Mstar=parameter(1,:)
beta_star=parameter(2,:)

xstar=Mstar
zstar=beta_star


%Find SSE for the Mstar and beta_star

F=Mstar./(1+(A.*exp(-beta_star*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE


% Prepare for the iteration

slope_Mnow=slope_M1
slope_betanow=slope_beta1
slope_Mpast=slope_M2
slope_betapast=slope_beta2

y_past=10000
```

```
y_now=L


%Section 5: Iteration


maxiteration=10000
stop=0.0000001



for iteration=1:1:maxiteration


if abs(y_now-y_past)>=stop

disp('You are using Quasi-Newton with Least Squares to estimate the S-Curve.')
iteration

y_past=y_now;

%Assign two more points
x_now=Mstar;
delta_M=0.1*x_now;
x_next=x_now-delta_M;

z_now=beta_star;
delta_beta=0.1*z_now;
z_next=z_now-delta_beta;

%Find SSE of x_next and z_next

A=(x_next/V(1,:))-1;
F=x_next./(1+(A.*exp(-z_next*T)));
Error=V-F;
SquaredError=Error.*Error;
SSE=sum(SquaredError);
L=SSE;

y_next=L;


%Find slope

slope_M=(y_next-y_now)/(x_next-x_now);
slope_beta=(y_next-y_now)/(z_next-z_now);
```

```
slope_Mnow=slope_M;
slope_betanow=slope_beta;

%Calculate vi and ui

vi= [x_next-x_now ; z_next-z_now ];
ui= [slope_Mnow-slope_Mpast ; slope_betanow-slope_betapast ] ;



%Calculate the next H

H=H + ((vi*vi')/(vi'*ui))-((H*ui*ui'*H)/(ui'*H*ui));

%Calculate d

g= [slope_Mpast ; slope_betapast];
d= (-H)*g;

%Calculate next M and beta

parameter= [ x_now ; z_now ];
parameter=parameter+d;

Mstar=parameter(1,:);
beta_star=parameter(2,:);

slope_Mpast=slope_M;
slope_betapast=slope_beta;

x_now=Mstar;
z_now=beta_star;

iteration

end

%Report

Mstar=x_now;
beta_star=z_now;
beta=beta_star;
iteration;
end


disp('Thank you very much for using KS Software and KSAN approach')
```

```
disp('-------------------------------------------------------------------------------')
disp('-------------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')

%----------------------------
%Calculate SSE (within model)



V=V2(:,1);
V=V(1:numbers_of_data,:);
T=data(:,2);
T=T(1:numbers_of_data,:);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));
SSE=sum((V-Vhat).^2);

N=length(V);
k=2;

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)


%Calculate MAPE for the out-of-sample-test

V=V2(:,1);
T=data(:,2);
N=length(V);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));

number_out_sample=32-numbers_of_data;

if number_out_sample>0

out_sample=V(32+1-number_out_sample:end,:);

Error=out_sample-Vhat(32+1-number_out_sample:end,:);
APE=abs(Error./out_sample)*100;
N=length(out_sample);
MAPE=sum(APE)./N

print=[ numbers_of_data Mstar beta  SSE MAPE  AIC  BIC ]

else
```

```
print=[ numbers_of_data Mstar beta SSE 0  AIC  BIC]

end
```

# Appendix F

## Matlab code for Gauss-Newton

```
% Analysis 6
% Quasi-Newton vs Gauss-Newton
% Sub-analysis 6.2
% Gauss-Newton
% Logistic model
% Least squares

clear

%Please specify the numbers of data in use.
numbers_of_data=32


%Please speicy three upper bounds of the logistic function "M3".
M3= [800000 ; 900000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000



%------------------------------------------------------------------


%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

load Sales

V=data(:,1)
T=data(:,2)

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3
safe = 99999

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
```

```
   end
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :)
grand_average=sum(monthly_average)/length(monthly_average)
seasonal_index=monthly_average*(100/grand_average)
s=seasonal_index
for r=2:1:year
   s = [s ; seasonal_index]
end
k=length(V)
s=s(1:k,:)
deseasonalized_V=V*100./(s)
V2=deseasonalized_V

%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000

%Calculate A to fix y=V0 at T=0

V=V2

V=V(1:numbers_of_data , :)
T=T(1:numbers_of_data , :)

A=(M/V(1,:))-1



% Now we are using Quasi-Newton to estimate both M and beta at the same time.
% We will fix y-intercept, therefore A is prior calculated.
% We will use Least Squares to be objective function.



%To find SSE and beta for the related M, we will use the OLS method.
%It is the efficient way to find beta from M, otherwise we guess beta from nothing.

results=[ 999 999 999]

for iteration=1:1:length(M3)
k=iteration
M=M3(k,:)
```

```
A=(M/V(1,:))-1
Y=log((V/M)./(1-(V/M)))-log(1/A)
beta=(inv(T'*T))*(T'*Y)
F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L beta ]
end

results=results(2:end,:)


% We extract all elements from the results.

x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

z1=results(1,3);
z2=results(2,3);
z3=results(3,3);


%Gauss-Newton

%Initial beta1
%beta2=beta1+ {inv(z'z)*z'(y-yhat)}
%Converged when beta2==beta1
%where
%y=real value
%yhat = forecasted
%z = [ delta_y(with all x of obs1)/delta first_beta ...  delta_y(with all x of obs1)/delta
last_beta ;
%      ...
%      delta_y(with all x of obs_N)/delta first_beta ; delta_y(with all x of obs_N)/delta
last_beta ]

% delta_y(with all x of obs1)/delta first_beta means delta of the first element of the
y_hat
% which are evaluated at two first_parameter (such as M1 and M2).
```

```
% delta_y(with all x of obs_N)/delta last_beta means delta of the last element of the
y_hat
% which are evaluated at two last_parameter (such as beta1 and beta2)



%-----------------------------------------------------------------
%I will use a pseudo data to test the algorithm as follows:


%V=M/1+Aexp(-T*beta)

%Step 1: Initial parameters

M1=x1  %Try M1=3000000 and M1=1000000
beta1=z1  %Try beta1=0.75 and beta1=0.25

%-----------------------------

M3=1.01*M1
MM= [  M1 ; M3 ]

beta3=1.1*beta1
BB= [ beta1 ; beta3 ]

results=[ T ]

%Step2: Repeat for all M and beta

for k=1:1:length(MM)
k
M=MM(k,:)
beta=BB(k,:)
A=(M/V(1,:))-1
F=M./(1+(A.*exp(-beta*T)))
results= [ results  F ]
end

results=results(: , 2:end)

% Calculate matrix z

MMM=MM'
BBB=BB'

delta_F=results(:,2)-results(:,1)
delta_M=MMM(:,2)-MMM(:,1)
```

```
delta_beta=BBB(:,2)-BBB(:,1)


zM=delta_F/delta_M
zB=delta_F/delta_beta

z=[ zM zB ]


%Calculate beta2=beta1+ {inv(z'z)*z'(y-yhat)}

a=z'*z
b=inv(z'*z)
yhat=results(:,1);
y=V;
d=b*z'*(y-yhat)
d=d'


M_next=M1+d(:,1)
beta_next=beta1+d(:,2)

percent_change=[(beta_next-beta1)/beta1 ; (M_next-M1/M1) ]


%Step 3: Repeats until convergence

maxiteration=6
iteration=1

%while max(percent_change)>0.1
%while iteration<maxiteration
while abs(beta_next - beta1)>0.01


M1=M_next
beta1=beta_next

M3=1.01*M1
MM= [  M1 ; M3 ]

beta3=1.1*beta1
BB= [ beta1 ; beta3 ]

results=[ T ]

%Step2: Repeat for all M and beta
```

```
for k=1:1:length(MM)
k
M=MM(k,:)
beta=BB(k,:)
A=(M/V(1,:))-1
F=M./(1+(A.*exp(-beta*T)))
results= [ results  F ]
end

results=results(: , 2:end)

% Calculate matrix z

MMM=MM'
BBB=BB'

delta_F=results(:,2)-results(:,1)
delta_M=MMM(:,2)-MMM(:,1)
delta_beta=BBB(:,2)-BBB(:,1)


zM=delta_F/delta_M
zB=delta_F/delta_beta

z=[ zM zB ]


%Calculate beta2=beta1+ {inv(z'z)*z'(y-yhat)}

a=z'*z
b=inv(z'*z)
yhat=results(:,1);
y=V;
d=b*z'*(y-yhat)
d=d'


M_next=M1+d(:,1)
beta_next=beta1+d(:,2)



iteration=iteration+1

end
```

```
% Summary
iteration
beta_final=beta_next
M_final=M_next


Mstar=M_final
beta=beta_final


disp('Thank you very much for using KS Software and KSAN approach')
disp('--------------------------------------------------------------------------------')
disp('--------------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')

%----------------------------
%Calculate SSE (within model)



V=V2(:,1);
V=V(1:numbers_of_data,:);
T=data(:,2);
T=T(1:numbers_of_data,:);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));
SSE=sum((V-Vhat).^2);

N=length(V);
k=2;

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)


%Calculate MAPE for the out-of-sample-test

V=V2(:,1);
T=data(:,2);
N=length(V);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));

number_out_sample=32-numbers_of_data;

if number_out_sample>0
```

```
out_sample=V(32+1-number_out_sample:end,:);

Error=out_sample-Vhat(32+1-number_out_sample:end,:);
APE=abs(Error./out_sample)*100;
N=length(out_sample);
MAPE=sum(APE)./N

print=[ numbers_of_data Mstar beta  SSE MAPE  AIC  BIC ]

else

print=[ numbers_of_data Mstar beta SSE 0  AIC  BIC]

end
```

## Appendix G

### Matlab code for Newton-Raphson

```
% Analysis 6
% Quasi-Newton vs Gauss-Newton
% Sub-analysis 6.3
% Newton-Raphson
% Logistic model
% Least squares

clear

%Please specify the numbers of data in use.
numbers_of_data=3

%Please speicy three upper bounds of the logistic function "M3".
M3= [800000 ; 900000 ; 1000000 ]  %These M3 are to be used in section 2.

%Please speciy the rounds of iteration
maxiteration=10000



%----------------------------------------------------------------


%Section 1: Data management and calculcation of likelihood function
%Step 1: Loading data

load Sales

V=data(:,1)
T=data(:,2)

%Step 2: Deseasonalization

%Please enter number of years of the data:
year=3
safe = 99999

for k=1:1:12  %12 months
   s=V(k,:)
for r=2:1:year  %years
   if 12*(r-1)+k <= length(V)
   s= [ s  V(12*(r-1)+k, :) ]
   end
```

```
end
   average=sum(s)/length(s)
   safe = [ safe ; average ]
end
monthly_average=safe(2:end, :)
grand_average=sum(monthly_average)/length(monthly_average)
seasonal_index=monthly_average*(100/grand_average)
s=seasonal_index
for r=2:1:year
   s = [s ; seasonal_index]
end
k=length(V)
s=s(1:k,:)
deseasonalized_V=V*100./(s)
V2=deseasonalized_V

%Step 3: Logistic transformation

%Please speicy the upper bound of the logistic function "M".
M=1000000

%Calculate A to fix y=V0 at T=0

V=V2

V=V(1:numbers_of_data , :)
T=T(1:numbers_of_data , :)

A=(M/V(1,:))-1


% Now we are using Quasi-Newton to estimate both M and beta at the same time.
% We will fix y-intercept, therefore A is prior calculated.
% We will use Least Squares to be objective function.


%To find SSE and beta for the related M, we will use the OLS method.
%It is the efficient way to find beta from M, otherwise we guess beta from nothing.

results=[ 999 999 999]

for iteration=1:1:length(M3)
k=iteration
M=M3(k,:)
A=(M/V(1,:))-1
```

```
Y=log((V/M)./(1-(V/M)))-log(1/A)
beta=(inv(T'*T))*(T'*Y)
F=M./(1+(A.*exp(-beta*T)))
Error=V-F
SquaredError=Error.*Error
SSE=sum(SquaredError)
L=SSE
results= [ results ; M L beta ]
end

results=results(2:end,:)


% We extract all elements from the results.

x1=results(1,1);
x2=results(2,1);
x3=results(3,1);

y1=results(1,2);
y2=results(2,2);
y3=results(3,2);

z1=results(1,3);
z2=results(2,3);
z3=results(3,3);


%Newton-Raphson

%Initial beta1
%beta2=beta1- inv(h)*delta_S/delta_beta
%inv(h) = slope of slope which is delta^2_S/delta_beta^2
% we can use S as SSE or Likelihood value



%---------------------------------------------------------------
%I will use a pseudo data to test the algorithm as follows:


%V=M/1+Aexp(-T*beta)


%Step 1: Initial parameters

M1=x1  %Try M1=3000000 and M1=1000000
```

```
beta1=z1  %Try beta1=0.75 and beta1=0.25

%-----------------------------

M2=0.9*M1
M3=1.1*M1
MM= [ M2 ; M1 ; M3 ]

beta2=0.9*beta1
beta3=1.1*beta1
BB= [ beta2 ; beta1 ; beta3 ]

Vhat=M1./(1+(A.*exp(-T*beta1)))

F=Vhat
e=V-F


results=[ 999 999 999 ]

%Step4: Repeat for all M and beta

for k=1:1:length(MM)
k
M=MM(k,:)
beta=BB(k,:)
A=(M/V(1,:))-1
F=M./(1+(A.*exp(-beta*T)))
L=sum((V-F).^2)
results= [ results ; M beta L ]
end

results=results(2:end, :)


%Step5: Calculate the slope

%Slope between M1 and M2

delta_L1=results(2,3)-results(1,3)
delta_M1=M1-M2
delta_beta1=beta1-beta2
slope_M1=delta_L1/delta_M1
slope_beta1=delta_L1/delta_beta1

%Slope between M2 and M3
delta_L2=results(3,3)-results(2,3)
```

```
delta_M2=M3-M1
delta_beta2=beta3-beta1
slope_M2=delta_L2/delta_M2
slope_beta2=delta_L2/delta_beta2


%Slope of slope

delta_slope_M=slope_M2-slope_M1
delta_slope_beta=slope_beta2-slope_beta1
sslope_M=delta_slope_M/(((M3+M1)/2)-((M1+M2)/2))
sslope_beta=delta_slope_beta/(((beta3+beta1)/2)-((beta1+beta2)/2))


%Step6: Iteration of Newton Raphson Algorithm


beta_next=beta1-(slope_beta1/sslope_beta)
M_next=M1-(slope_M1/sslope_M)

result_next= [ beta_next ; M_next ]
result_old=[ beta1 ; M1]

percent_change=[(beta_next-beta1)/beta1 ; (M_next-M1/M1) ]

L_old=999
L1=results(2,3)
L_now=L1


%Step 7: Repeats until convergence

iteration=1

%while (abs(L_now-L_old)) > 0.00000000000000000001*L_now
for kkk=1:1:maxiteration

disp ('You are using Newton_Raphson to estimate the S-Curve.')
results= [ 999 999 999 ];
L_old=L_now;

%Initial M and beta again
M1=M_next;
beta1=beta_next;


M2=0.9*M1;
```

```
M3=1.1*M1;
MM= [ M2 ; M1 ; M3 ];

beta2=0.9*beta1;
beta3=1.1*beta1;
BB= [ beta2 ; beta1 ; beta3 ];


%Calculate likelihood value

%Probability density function (p.d.f.)

%Assume normal p.d.f.
% f(Vi) = 1/sigma*sqrt(2Pi) * exp (-1/2)*[(Vi - Vhat,i)/sigma]^2


pi=3.14159265359;
sigma = 20000;

%Repeat for all M and beta


for k=1:1:length(MM)
k;
M=MM(k,:)  ;
beta=BB(k,:);
A=(M/V(1,:))-1;
F=M./(1+(A.*exp(-beta*T)));
L=sum((V-F).^2);
results= [ results ; M beta L ];
end

results=results(2:end, :);


%Calculate the slope

%Slope between M1 and M2

delta_L1=results(2,3)-results(1,3);
delta_M1=M1-M2;
delta_beta1=beta1-beta2;
slope_M1=delta_L1/delta_M1;
slope_beta1=delta_L1/delta_beta1;

%Slope between M2 and M3
delta_L2=results(3,3)-results(2,3);
```

```
delta_M2=M3-M1;
delta_beta2=beta3-beta1;
slope_M2=delta_L2/delta_M2;
slope_beta2=delta_L2/delta_beta2;


%Slope of slope

delta_slope_M=slope_M2-slope_M1;
delta_slope_beta=slope_beta2-slope_beta1;
sslope_M=delta_slope_M/(((M3+M1)/2)-((M1+M2)/2));
sslope_beta=delta_slope_beta/(((beta3+beta1)/2)-((beta1+beta2)/2));


%Step6: Iteration of Newton Raphson Algorithm


beta_next=beta1-(slope_beta1/sslope_beta);
M_next=M1-(slope_M1/sslope_M);

result_next= [ M_next ; beta_next ];
result_old=[ M1 ; beta1];

L1=results(2,3);
L_now=L1;

percent_change=[(M_next-M1/M1); (beta_next-beta1)/beta1  ];

iteration=iteration+1;

end
% Summary
iteration;
beta_final=beta_next;
M_final=M_next;



Mstar=M_final;
beta=beta_final;


disp('Thank you very much for using KS Software and KSAN approach')
disp('-----------------------------------------------------------------------------')
disp('-----------------------------------------------------------------------------')
disp('This version limits that M cannot be lower than the larger value of V')
disp('such that V/M cannot exceed 1. Otherwise, the model will not work.')
```

```
%-----------------------------
%Calculate SSE (within model)



V=V2(:,1);
V=V(1:numbers_of_data,:);
T=data(:,2);
T=T(1:numbers_of_data,:);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));
SSE=sum((V-Vhat).^2);

N=length(V);
k=2;

AIC=log(SSE./N)+(2*k./N)
BIC=log(SSE./N)+(k.*log(N)./N)


%Calculate MAPE for the out-of-sample-test

V=V2(:,1);
T=data(:,2);
N=length(V);

Vhat=Mstar./(1+(A./(1+exp(-beta*T))));

number_out_sample=32-numbers_of_data;

if number_out_sample>0

out_sample=V(32+1-number_out_sample:end,:);

Error=out_sample-Vhat(32+1-number_out_sample:end,:);
APE=abs(Error./out_sample)*100;
N=length(out_sample);
MAPE=sum(APE)./N

print=[ numbers_of_data Mstar beta  SSE MAPE  AIC  BIC ]

else

print=[ numbers_of_data Mstar beta SSE 0  AIC  BIC]

end
```

# Appendix H

## Matlab code for KS-CGE Type IV

```
%KS CGE Version 2012 Type IV
%Gauss-Seidel Iteration
%CES Technology
%Handle N*N matrix
%(c) Komsan Suriya
%Chiang Mai School of Economics, Thailand
%28 November 2012
%Special edition for the thesis of Orakanya Kanjanatarakul

clear

load CGE_RPF_new2
%xy=data


%Settings
firm=16
importers=14
household=10
institution=1
government=1
margin=1  %Transaction cost (wholesale, retail, transportation cost)
tax=1

sector=firm+importers+household+institution+government+margin+tax
s=sector

sigma=0.01*ones(s,s)
sigma(:,s)=0  % Tax

gauss_seidel_maxiteration=10
model_maxiteration=10

%Extract data matrices
x=xy(:,1:s)
y=xy(:,s+1)

%Verification of balanced data matrix
if det(x)==0 %Data matrix verification
   disp('Good! Go ahead. det (x) is zero.')
else
```

```
   disp('Warning! det(x)is not zero. The data matrix is unbalanced.')
end

valid=999
for k=1:1:s
   val=sum(x(:,k))
   valid = [ valid ; val]
end

valid=valid(2:s+1,:)
if sum(valid)==0
   disp('Good! Domestic sales are balanced.')
else
   disp('Warning! Unbalanced domestic sales.')
end




%Section I: Gauss-Seidel Iteration

dx=diag(x);
xc=x-diag(dx);
yy=(diag(y));
qd=-xc+yy;
div=dx*ones(1,s);
qdiv=qd./div;


initial_price=1*ones(s,1);  %For model validation purpose
intp=initial_price;

%Condition to start iteration
last_p=intp;
p=intp+1;

%Counter
iteration=0;

%Gauss-Seidel Iteration


while sum(abs(p-last_p))>0.0001
   last_p=p;

if iteration<=gauss_seidel_maxiteration
```

```
for k=1:1:s


   dintp=intp*ones(1,s);
   dintpp=diag(dintp);
   pp=dintp-diag(dintpp)+eye(s);
   pp(:,k);
   shot=qdiv(k,:)*pp(:,k);
   intp(k,:)=shot;
   p=intp;


end

%Counter
   iteration=iteration+1
end
end

finalp=p  %when initial price =1 , this p must be 1 (benchmark)
lastp=p-0.1*p;  %Starting condition for WHILE iteration

xx=x;

if prod(round(finalp))==1
   disp('Benchmarked prices are all one. OK! go ahead.')
else
   disp('Prices change because of the counterfactual.')
end


%--------------------------------------------------------------%
%%Iteration of CGE modelling until all prices are converged.

rounds=0


%Option 1: to fix the maxround
maxround=5  %Must select maxround in order to calibrate the model
for r=1:1:maxround

%Option 2: to find the convergence.
%while sum(abs(finalp-lastp))>(s*0.01)

rounds=rounds+1
```

```
disp('You are running CGE model of Chiang Mai School of Economics.')
lastp=finalp;
x=xx    ;


%%--------------------------------------------------------------
%---------------------------------------------------------------%

%Section II: Shephard's lemma
%Now the input ratio would be changed.

%Find total cost
domsales=diag(x);
intertrade=-y;
totalsales=domsales+intertrade;
totalcost=totalsales;

%Find alpha
%Alpha_ij=(input from seller i to customer j)*{(totalcost of customer j)^(-sigma)}

dx=diag(x);
xc=x-diag(dx);
tc=totalcost*ones(1,s);
alpha=abs(xc).*(tc.^(-sigma));


%Selection of sigma at position i j


xx=x;
repeat=0;
redeam=0;

for i=1:1:length(sigma)     %Selling firm
   for j=1:1:length(sigma) %Buying firm


   if x(j,i)~=0  %For all non-zero inputs
      if j~=i
      sigmaij=sigma(j,i);

      f1=finalp.^(1-sigmaij);
      f2=alpha(j,i).*f1;
      f3=sum(f2(:,1)).^(sigmaij/(1-sigmaij));
      f4=alpha(j,i)*(finalp(i,:).^-sigmaij);
      xx(j,i)=-(f4*f3);
```

```
        %Counter
        repeat=repeat+1;
        redeam=redeam+1;
        end

        if j==i
        xx(j,i)=0;
        end
    end

    if x(j,i)==0   %When firm does not use this input, it must not use it in next round.
        xx(j,i)=0  ;
    end

    end
end

xx;

for i=1:1:length(sigma)     %Selling firm
    for j=1:1:length(sigma) %Buying firm

        if j==i
        xx(j,i)=-sum(xx(:,i));

        %Counter
        repeat=repeat+1;
        redeam=redeam+1;
        end
    end
end

xx;
x;
repeat;
redeam;


%Section III: Find new price
%Gauss-Seidel Iteration

x=xx;

dx=diag(x);
xc=x-diag(dx);
yy=(diag(y));
qd=-xc+yy;
```

```
div=dx*ones(1,s);
qdiv=qd./div;


initial_price=finalp;
intp=initial_price;

%Condition to start iteration
last_p=intp;
p=intp+1;

%Counter
g_iteration=0;


%Gauss-Seidel Iteration


while sum(abs(p-last_p))>0.0001
   last_p=p;
if g_iteration<=gauss_seidel_maxiteration
for k=1:1:s


   dintp=intp*ones(1,s);
   dintpp=diag(dintp);
   pp=dintp-diag(dintpp)+eye(s);
   pp(:,k);
   shot=qdiv(k,:)*pp(:,k);
   intp(k,:)=shot;
   p=intp;
   finalp=p;

end

%Counter
   g_iteration=g_iteration+1;
end
end
end


echo on
%The initial world
echo off
xy(1:s,1:s)
```

```
echo on
%The world after impact
echo off
xx

finalp  %when initial price =1 , this p must be 1 (benchmark)

echo on
%Growth of each sector
echo off

growth=(diag(xx).*finalp)./diag(xy)
rounds

for t=1:1:length(growth)
   if growth(t,:)<0
      disp('Warning! Negative number in growth matrix which indicates the collapse of the sector.')
   else
      disp('Checked, OK.')
   end
end


calibrate=growth;
calibrate_p=finalp;

finalp=finalp./calibrate_p;
growth=growth./calibrate

%%---------------End of calibration---------------------------------


%%%%-------------The Model with counterfactual-----------------------
%%-----------------------------------------------------------
%%-----------------------------------------------------------
%%Counterfactual

%Extract data matrices
x=xy(:,1:s)
y=xy(:,s+1)

echo on

%(1): Taxation
g=1.00;
```

```
rowx=44;
columnx=44;
x(rowx,columnx)=g*x(rowx,columnx);



%(2) Labour cost

wage=1.00
salary=1.00
benefit=1.00

row_hh1=31;
row_last_hh=row_hh1+household-1;
xl=diag(xy);
xl=xl(row_hh1:row_last_hh, :);

lower_xl=xl(1:round(0.4*household),:);
middle_xl=xl(round(0.4*household)+1:round(0.8*household),:);
higher_xl=xl(round(0.8*household)+1:household,:);

lower_xl=wage*lower_xl;
middle_xl=salary*middle_xl;
higher_xl=benefit*higher_xl;

dxl= [ lower_xl ; middle_xl ; higher_xl ];

for v=0:1:household-1
x(row_hh1+v,row_hh1+v)=dxl(v+1,:);
end



%(3) Intermediate input cost

g_sector1=1.10
g_sector2=1.00
g_sector3=1.10
g_sector4=1.00
g_sector5=1.00
g_sector6=1.00
g_sector7=1.00
g_sector8=1.00
g_sector9=1.00
g_sector10=1.00
g_sector11=1.00
g_sector12=1.00
```

```
g_sector13=1.00
g_sector14=1.00
g_sector15=1.00
g_sector16=1.00
%g_sector17=1.00

g_firm = [ g_sector1 ; g_sector2 ; g_sector3 ; g_sector4 ;
         g_sector5 ; g_sector6 ; g_sector7 ; g_sector8 ;
         g_sector9 ; g_sector10 ; g_sector11; g_sector12 ;
         g_sector13 ; g_sector14 ; g_sector15 ; g_sector16 ]


for r=1:1:firm

   x(r,r)=g_firm(r,:)*x(r,r)

end


%(4): Money supply
moneysupply=1.00

x=moneysupply*x;


%%---This effect is too much.------------------
%for r=1:1:length(x)
%x(r,r)=xy(r,r);  %Price of each sector's output does not increase at the beginning.
%end
%%Better let the price of output grows up with the expandary money supply.
%%---------------------------------------------


%(5): Capital inflow (and government subsidy by cash) just for some sectors

addedmoney=2.225
sect=16
govsec=firm+importers+household+institution+government
govfunding=xy(govsec,sect)
a=-(sum(x(sect,:))-x(sect,sect))
b=(-(addedmoney-1)*govfunding)
c=a+b
add=c/a
x(sect,:)=add*x(sect,:);

% This effect is also too much.
%x(sect,sect)=xy(sect,sect);
```

```
%(6) External trade

t_sector1=1.00
t_sector2=1.00
t_sector3=1.00
t_sector4=1.00
t_sector5=1.00
t_sector6=1.00
t_sector7=1.00
t_sector8=1.00
t_sector9=1.00
t_sector10=1.00
t_sector11=1.00
t_sector12=1.00
t_sector13=1.00
t_sector14=1.00
t_sector15=1.00
t_sector16=1.00
%t_sector17=1.00

t_firm = [ t_sector1 ; t_sector2 ; t_sector3 ; t_sector4 ;
        t_sector5 ; t_sector6 ; t_sector7 ; t_sector8 ;
        t_sector9 ; t_sector10 ; t_sector11; t_sector12 ;
        t_sector13 ; t_sector14 ; t_sector15 ; t_sector16 ]

for r=1:1:firm

   y(r,1)=t_firm(r,1)*y(r,1)

end


% (7) Change of household consumption for some sectors

g_sector=1.00

sect=16 %Selling sector

first_hh_percentile=5  %change from 1 - 10 (1 is poorest and 10 is richest)
end_hh_percentile=10   %change from 1 - 10
firstrow_hh=31
```

```
for r=first_hh_percentile-1:1:end_hh_percentile-1
   x(firstrow_hh+r,sect)=g_sector*x(firstrow_hh+r,sect)
end


% (8) Change of government budget (overall)

g=1.00

rowx=42
x(rowx,:)=g*x(rowx,:);  % Gov pays more to other sectors.
x(rowx,rowx)=xy(rowx,rowx);  %Cost of gov commerce does not increase. Otherwise
fiscal expansion will shrinken the economy.


% (9) Government spending into some sectors

%You can compare the subsidy paid to two sectors at the same time.

g1=1.00
sect1=16

g2=1.00
sect2=3


rowx=42
x(rowx,sect1)=g1*x(rowx,sect1);
x(rowx,sect2)=g2*x(rowx,sect2);

%Note: If the subsidy is switched from a constant gov budget,
%you just use the counterfactual in (9).
%If the subsudy is financed by the expansion of gov budget,
%you need to add counterfactual (8).


%end of counterfactual
%%-------------------------------------------------------------
%%-------------------------------------------------------------
echo off


%Section I: Gauss-Seidel Iteration
```

```
dx=diag(x);
xc=x-diag(dx);
yy=(diag(y));
qd=-xc+yy;
div=dx*ones(1,s);
qdiv=qd./div;


initial_price=1*ones(s,1);  %For model validation purpose
intp=initial_price;

%Condition to start iteration
last_p=intp;
p=intp+1;

%Counter
iteration=0;

%Gauss-Seidel Iteration


while sum(abs(p-last_p))>0.0001
   last_p=p;

if iteration<=gauss_seidel_maxiteration

for k=1:1:s



   dintp=intp*ones(1,s);
   dintpp=diag(dintp);
   pp=dintp-diag(dintpp)+eye(s);
   pp(:,k);
   shot=qdiv(k,:)*pp(:,k);
   intp(k,:)=shot;
   p=intp;


end

%Counter
   iteration=iteration+1
end
end

finalp=p  %when initial price =1 , this p must be 1 (benchmark)
```

```
lastp=p-0.1*p;  %Starting condition for WHILE iteration

xx=x;

if prod(round(finalp))==1
   disp('Benchmarked prices are all one. OK! go ahead.')
else
   disp('Prices change because of the counterfactual.')
end


%--------------------------------------------------------------%
%%Iteration of CGE modelling until all prices are converged.

rounds=0


%Option 1: to fix the maxround
maxround=5  %Must select maxround in order to calibrate the model
for r=1:1:maxround

%Option 2: to find the convergence.
%while sum(abs(finalp-lastp))>(s*0.01)

rounds=rounds+1

disp('You are running CGE model of Chiang Mai School of Economics.')
lastp=finalp;
x=xx    ;


%%--------------------------------------------------------------
%---------------------------------------------------------------%

%Section II: Shephard's lemma
%Now the input ratio would be changed.

%Find total cost
domsales=diag(x);
intertrade=-y;
totalsales=domsales+intertrade;
totalcost=totalsales;

%Find alpha
%Alpha_ij=(input from seller i to customer j)*{(totalcost of customer j)^(-sigma)}

dx=diag(x);
```

```
xc=x-diag(dx);
tc=totalcost*ones(1,s);
alpha=abs(xc).*(tc.^(-sigma));


%Selection of sigma at position i j


xx=x;
repeat=0;
redeam=0;

for i=1:1:length(sigma)     %Selling firm
    for j=1:1:length(sigma) %Buying firm


    if x(j,i)~=0  %For all non-zero inputs
        if j~=i
        sigmaij=sigma(j,i);

        f1=finalp.^(1-sigmaij);
        f2=alpha(j,i).*f1;
        f3=sum(f2(:,1)).^(sigmaij/(1-sigmaij));
        f4=alpha(j,i)*(finalp(i,:).^-sigmaij);
        xx(j,i)=-(f4*f3);

        %Counter
        repeat=repeat+1;
        redeam=redeam+1;
        end

        if j==i
        xx(j,i)=0;
        end
    end

    if x(j,i)==0   %When firm does not use this input, it must not use it in next round.
        xx(j,i)=0  ;
    end

    end
end

xx;

for i=1:1:length(sigma)     %Selling firm
    for j=1:1:length(sigma) %Buying firm
```

```
        if j==i
        xx(j,i)=-sum(xx(:,i));

        %Counter
        repeat=repeat+1;
        redeam=redeam+1;
        end
    end
end

xx;
x;
repeat;
redeam;


%Section III: Find new price
%Gauss-Seidel Iteration

x=xx;

dx=diag(x);
xc=x-diag(dx);
yy=(diag(y));
qd=-xc+yy;
div=dx*ones(1,s);
qdiv=qd./div;


initial_price=finalp;
intp=initial_price;

%Condition to start iteration
last_p=intp;
p=intp+1;

%Counter
g_iteration=0;


%Gauss-Seidel Iteration


while sum(abs(p-last_p))>0.0001
    last_p=p;
if g_iteration<=gauss_seidel_maxiteration
```

```
for k=1:1:s


    dintp=intp*ones(1,s);
    dintpp=diag(dintp);
    pp=dintp-diag(dintpp)+eye(s);
    pp(:,k);
    shot=qdiv(k,:)*pp(:,k);
    intp(k,:)=shot;
    p=intp;
    finalp=p;

end

%Counter
    g_iteration=g_iteration+1;
end
end
end




echo on
%The initial world
echo off
xy(1:s,1:s)

echo on
%The world after impact
echo off
xx

finalp  %when initial price =1 , this p must be 1 (benchmark)

echo on
%Growth of each sector
echo off

growth=(diag(xx).*finalp)./diag(xy);



%Validate the results.
for t=1:1:length(growth)
```

```
   if growth(t,:)<0
      disp('Warning! Negative number in growth matrix which indicates the collapse of the sector.')
   else
      disp('Checked, OK.')
   end
end



disp('Congratulations! You have achieved the CGE result.')
disp('------------------------------------------------')
disp('Thank you for choosing KS software.')
disp ('(c) Komsan Suriya, 2012')


echo on
%Reports
echo off

rounds

growth_final=growth./calibrate

finalp=finalp./calibrate_p;

quant_final=growth_final./finalp;

inflation=(diag(xy)'*finalp)/sum(diag(xy))

gdp_growth=(diag(xy)'*growth_final)/sum(diag(xy))

%Economic sectors
sector_growth=growth_final(1:firm, :);
sector_price=finalp(1:firm, :);
sector_quant=quant_final(1:firm, :);

%Households
hh=firm+importers;
hh_growth=growth_final(hh+1:hh+household, :);
hh_price=finalp(hh+1:hh+household, :);
hh_quant=quant_final(hh+1:hh+household, :);

Print = [ growth_final ; gdp_growth ; inflation ]


%%---------------End of model------------------------------------------
```

# Curriculum Vitae

**Name** Ms. Orakanya Kanjanatarakul

**Date of Birth** 11th November, 1982

**Education Background** Master of Economics, Chiang Mai University, Chiang Mai, Thailand, 2007
Bachelor of Science (Product Development and Technology), Chiang Mai University, Chiang Mai, Thailand, 2005

**Scholarships** PhD scholarship from Office of the Higher Education Commission, Ministry of Education, Thailand

**Working Experiences** Lecturer, Faculty of Business Administration and Liberal Arts, Rajamangala University of Technology Lanna Chiang Mai (2012)