Lesson 8
Fixed Point blockset

Instructor:
ir drs E.J Boks
Elecrical Engineering
Embedded Systems Engineering
phone:(026) 3658173
Room: D2.03
e-mail:ewout.boks@han.nl

Microprocessors

- Microprocessors always process data in set chunks of memory. This set chunk of memory is called the processing size. The processing size is always a multiple of 2.
- Examples:
  - Atmel AVR microcontroller: 8 bits
  - Intel 80186 : 16 bits
  - Motorola DSP56000: 24 bits
  - ARM 7/9/11 core: 32 bits
  - Texas Instruments TMS320C6000: 128 bits

Infinite vs finite arithmetic

- Analog signals exist in inifite form, ie the analog representation has an inifite resolution. When processing signals in a digital system, these must be sampled into a finite resolution.
- Take for example: $V_{signal} = 2.587$ V
- This number can be represented in finite form by a number of (binary) digits.
- Range of the finite form for a binary number with n digits: $2^n - 1$ possible numbers

Infinite vs finite arithmetic

- The higher the amount of digits, the greater the range (with fixed resolution) or the resolution (with fixed range) :
- 4 digits
  - range = 0 .. 15.
  - 4 digits --> resolution = 5/16 = 0.31 V/step
  - 2.587 V represented as Round[(2.587/5)*16]*5/16 = 2.5000 V
- 12 digits
  - 12 digits --> range = 0 .. 4095
  - 12 digits --> resolution = 5/4096 = 1.22E-3 V/step
  - 2.587 V represented as Round[(2.587/5)*4096]*5/4096 = 2.5866 V
- 32 digits
  - 32 digitis --> range = 0 .. 4295*10^6
  - 32 digits --> resolution = 5/4295E6 = 1.16E-9 V/step
  - 2.587 V represented as Round[(2.587/5)*4295E6]*5/4295E6 = 2.5870 V

Fixed point versus floating point

- Modern signal processing engineers need to trade off between speed and accuracy, in order to keep checks on other criteria such as cost and feasibility.
- There are two principal means of digitally representing a signal in a microprocessor:
  - A signal representation using IEEE floating point means that a signal is represented using two integer locations: < fixed value> . <floating value>
  - A signal representation using fixed point value means that a signal is represented using only one integer location.

Floating point representation seems much better suited to helping solve engineering tasks such as control engineering. Why bother with fixed point?

Fixed point advanages

- *Size and Power Consumption* -- The logic circuits of fixed-point hardware are much less complicated than those of floating-point hardware. This means that the fixed-point chip size is smaller with less power consumption when compared with floating-point hardware. For example, consider a portable telephone where one of the product design goals is to make it as portable (small and light) as possible. If one of today's high-end floating-point, general-purpose processors is used, a large heat sink and battery would also be needed resulting in a costly, large, and heavy portable phone.
- *Memory Usage and Speed* -- In general fixed-point calculations require less memory and less processor time to perform.
- *Cost* -- Fixed-point hardware is more cost effective where price/cost is an important consideration. When using digital hardware in a product, especially mass-produced products, fixed-point hardware costs much less than floating-point hardware and can result in significant savings.

Requirements for employing fixed point arithmetic

- In order to utilise the range/resolution of a fixed point integer most efficiently, *scaling* the signal to match the integer characteristics is necessary.
- Without scaling, an integer signal is prone to:
  - Overflow
  - Bad representation, i.e unecessary loss of resolution or resolution overkill by using too large integer representations

Scaling an integer fixed point value

- Coefficients in signal processing are often floating point values. These cannot be represented by an integer value.
- By means of a multiplication, such that the largest processed number still does not cause an integer overflow, we scale or map a (floating point) value across an integer range.
- Example: the fractional value of 1.0000 is the highest number in our range, which will be mapped across a 16 bit integer range. We need this to specify a fractional value of 0.9001
  - By multiplying the number with 2^{16}, we scale everything between 0 and 1.0000 as an integer number between 0 and 65535.
  - The number 0.9001 is then mapped as 58988.9536 or (as integer) 58989. The rounding error is (1-0.9536)/2E16 = 7.08E-7 .
- Scaling is always done using powers of 2. This has the advantage that scaling can be done by shifting bits right and left, a very simple and fast microprocessor operation.

Determining integer size with regards to overflow

- To avoid integer overflow, keep this in mind:
  - Multiplying two N-digit numbers *can* results in a 2N digit number.
  - Summing two N-digit numbers *can* result in a N+1 digit number. Summing M numbers can result in a $N+\log_2(M)$ number.
- Example: a 10 bit Analogue to Digital Converter signal is multiplied with a 16-bit scaled filter coefficient in a 5th order FIR filter.
- The filter result needs a integer location of the size N:
  - $N = 10 + 16 + \log_2(5) = 28.3 = 29$ bits

Fixed Point Blockset overview

The Fixed-Point Blockset includes a collection of blocks that extend the standard Simulink block library. With these blocks, you can create discrete-time dynamic systems that use fixed-point arithmetic. As a result, Simulink can simulate effects commonly encountered in fixed-point systems for applications such as control systems and time-domain filtering.

Fixed point blockset overview (2)

- Integer, fractional, and generalized fixed-point data types
  - Unsigned and two's complement formats
  - Word sizes in simulation from 1 to 128 bits
- Floating-point data types
  - IEEE-style singles and doubles
  - A nonstandard IEEE-style data type, where the fraction can range from 1 to 52 bits and the exponent can range from 1 to 11 bits
- Methods for overflow handling, scaling, and rounding of fixed-point data types
- Tools that facilitate
  - The collection of minimum and maximum simulation values
  - The optimization of scaling parameters
  - The display of input and output signals

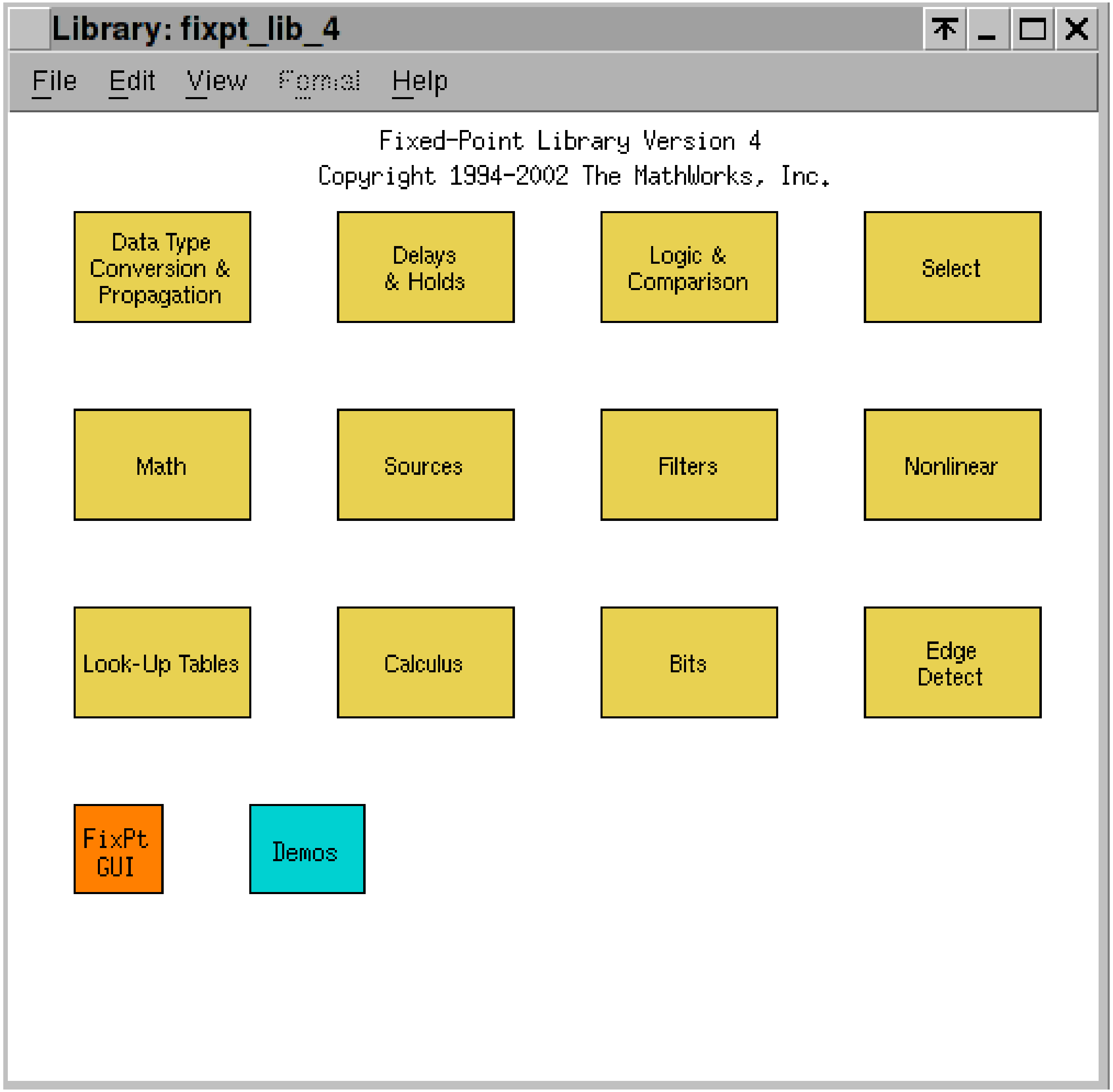
Fixed Point Blockset overview (3)

- In addition, you can generate C code for execution on a fixed-point embedded processor with the additionally purchased Real-Time Workshop.
- The generated code uses only integer types and automatically includes all operations, such as shifts, needed to account for differences in fixed-point locations.

Starting the blockset

- To start the blockset, type fixpt at the cmd prompt or locate the fixed point blockset in the library browser.

Blockset library



Fixed-Point Library Version 4
Copyright 1994-2002 The MathWorks, Inc.

Bits

- *Bit Clear* : Set the specified bit of the stored integer to zero
- *Bit Set* : Set the specified bit of the stored integer to one
- *Bitwise Operator* : Perform the specified bitwise operation on the inputs
- *Shift Arithmetic* : Arithmetically shift the bits and/or the radix point of a signal

Calculus : terminology used in the fixed point blockset

- *Accumulator* : compute a cumulative sum
- *Accumulator Resettable* : compute a cumulative sum with external Boolean reset
- *Accumulator Resettable Limited* : compute a limited cumulative sum with external Boolean reset
- *Derivative* : compute a discrete time derivative
- *Difference* : calculate the change in a signal over one time step
- *Integrator Backward* : perform discrete-time integration of a signal using the backward method

Calculus (2)

- *Integrator Backward Resettable* : perform discrete-time integration of a signal using the backward method, with external Boolean reset
- *Integrator Backward Resettable Limited* : perform discrete-time limited integration of a signal using the backward method, with external Boolean reset
- *Integrator Forward* : perform discrete-time integration of a signal using the forward method
- *Integrator Forward Resettable* : perform discrete-time integration of a signal using the forward method, with external Boolean reset

Calculus (3)

- *Integrator Forward Resettable Limited* :erform discrete-time limited integration of a signal using the forward method, with external Boolean reset
- *Integrator Trapezoidal* : perform discrete-time integration of a signal using the trapezoidal method
- *Integrator Trapezoidal Resettable* : perform discrete-time integration of a signal using the trapezoidal method, with external Boolean reset
- *Integrator Trapezoidal Resettable Limited* : perform discrete-time limited integration of a signal using the trapezoidal method, with external Boolean reset

Calculus (4)

- *Sample Rate Probe* : output weighted sample rate
- *Sample Time Add* : add the input signal to weighted sample time
- *Sample Time Divide* : divide the input signal by weighted sample time
- *Sample Time Multiply* : multiply the input signal by weighted sample time
- *Sample Time Probe* : output weighted sample time
- *Sample Time Subtract* : subtract weighted sample time from the input signal

Data type conversion

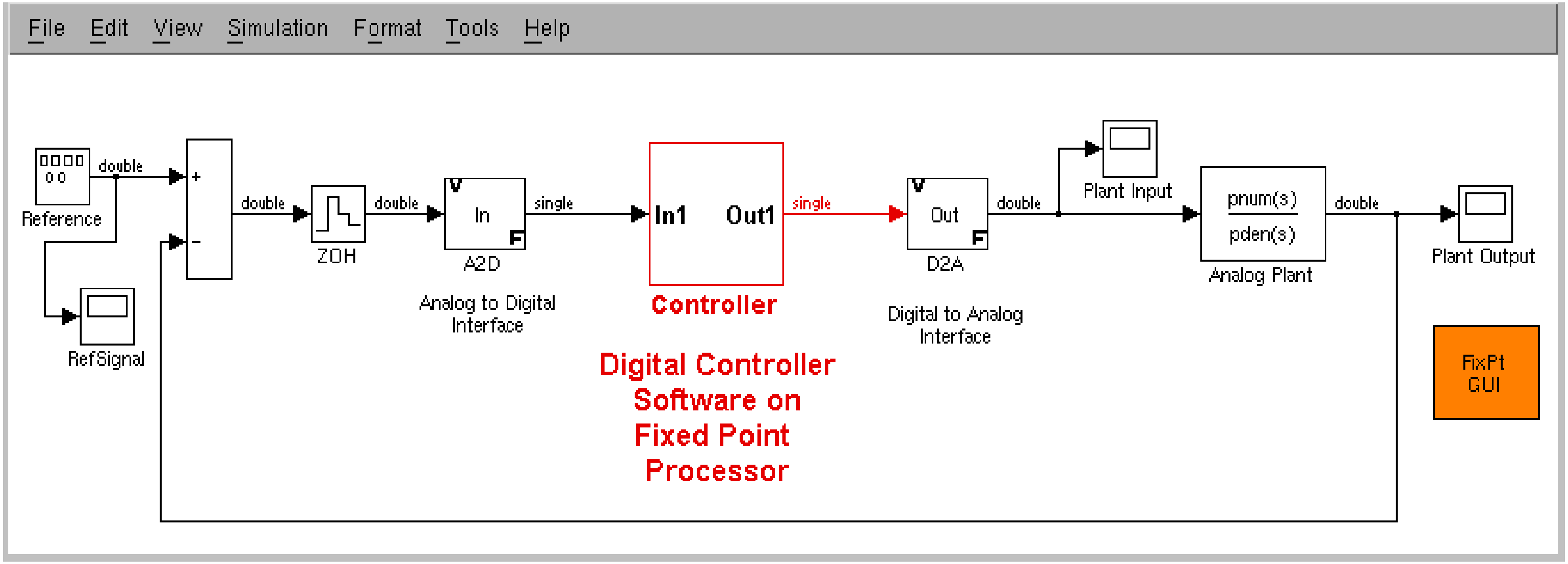
- *Conversion* : convert from one Fixed-Point Blockset data type to another
- *Conversion Inherited* : convert from one Fixed-Point Blockset data type to another, and inherit the data type and scaling
- *Data Type Duplicate* : set all inputs to the same data type
- *Data Type Propagation* : configure the data type and scaling of the propagated signal based on information

Data type conversion

- *Gateway In* : convert a Simulink data type to a Fixed-Point Blockset data type
- *Gateway In Inherited* : convert a Simulink data type to a Fixed-Point Blockset data type, and inherit the data type and scaling
- *Gateway Out* : convert a Fixed-Point Blockset data type to a Simulink data type
- *Scaling Strip* : remove scaling and map to a built-in integer

Example: A digital feedback controller

- Simulate a feedback control system where the controller is a 16 bit microprocessor.
- The system overview:



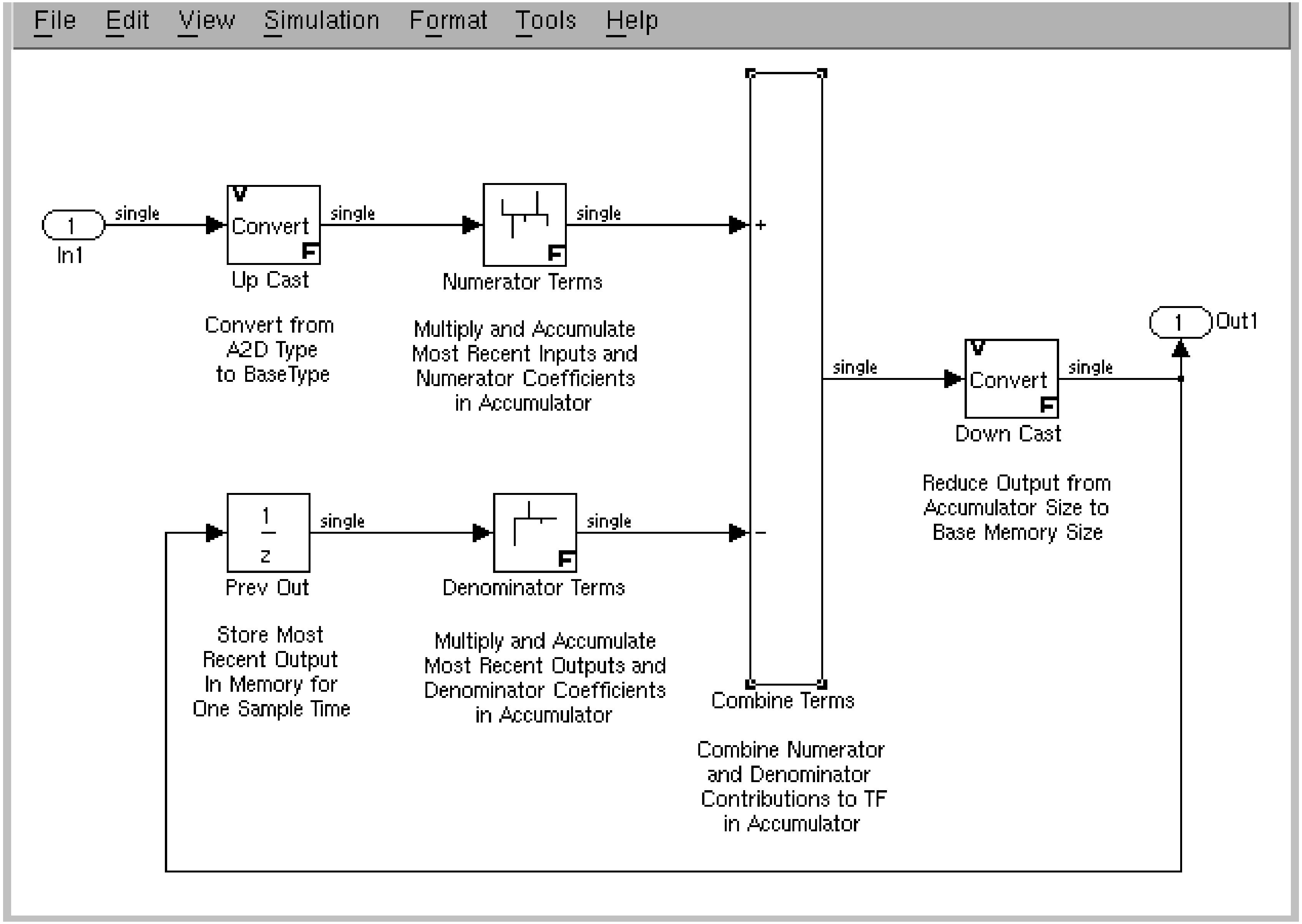
Modelling the Analogue to Digital converter (ADC)

- To simulate a real-world ADC, a Gateway In block is used. This block converts a Simulink double to a Fixed-Point Blockset data type.
- In the real world, its characteristics are fixed. However, in Simulink we can alter its characteristics.

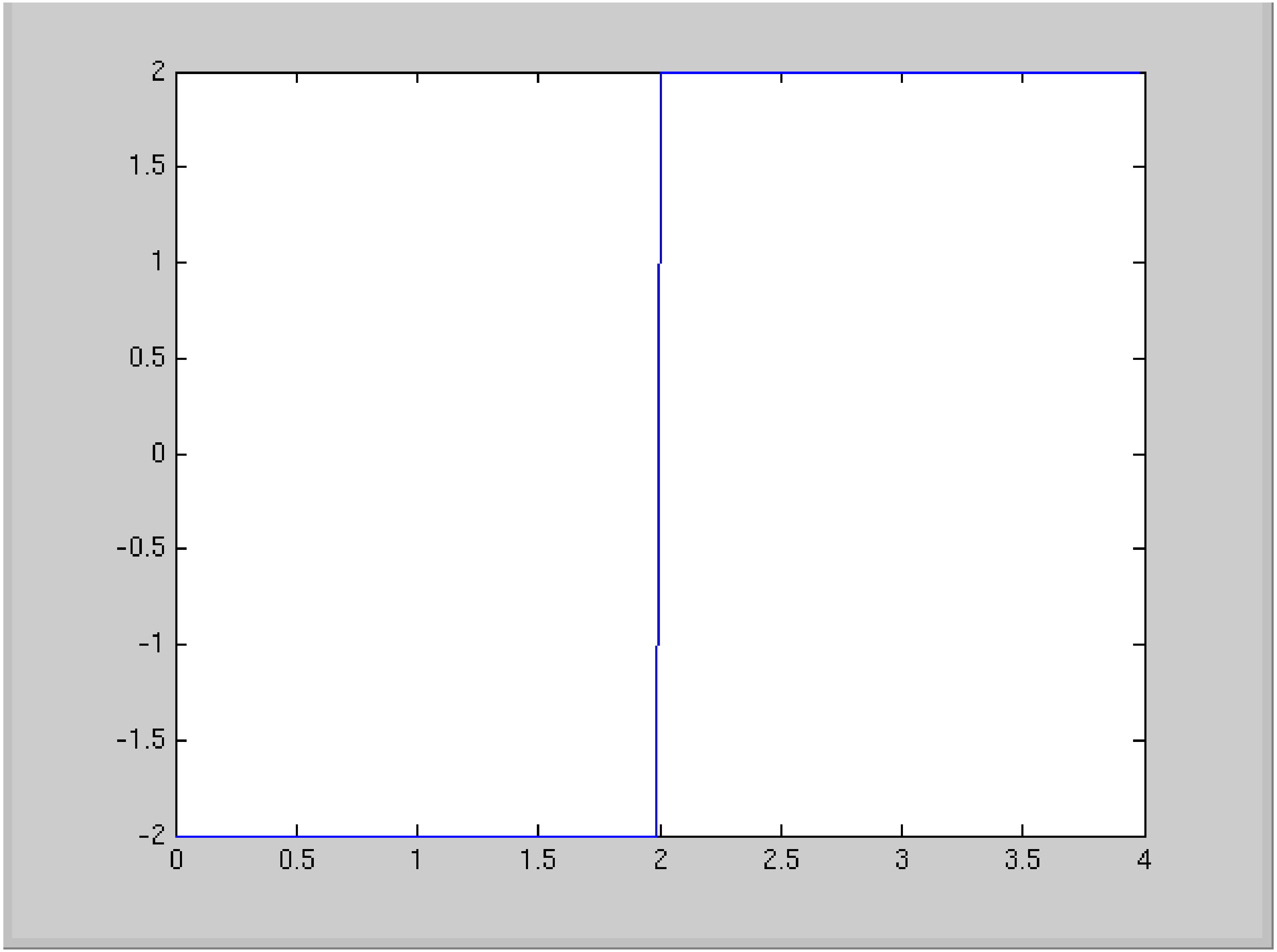
The controller

- The controller model must meet the following criteria:
  - The hardware target is a 16-bit processor.
  - Variables and coefficients are generally represented using 16 bits, especially if these quantities are stored in ROM or global RAM.
  - Use of 32-bit numbers is limited to temporary variables that exist briefly in CPU registers or in a stack.
- Implementation of the feedback mechanism follows a Z transform of the desired output behaviour. The transfer function consists therefore of a numerator (for the input) and a denominator (for the

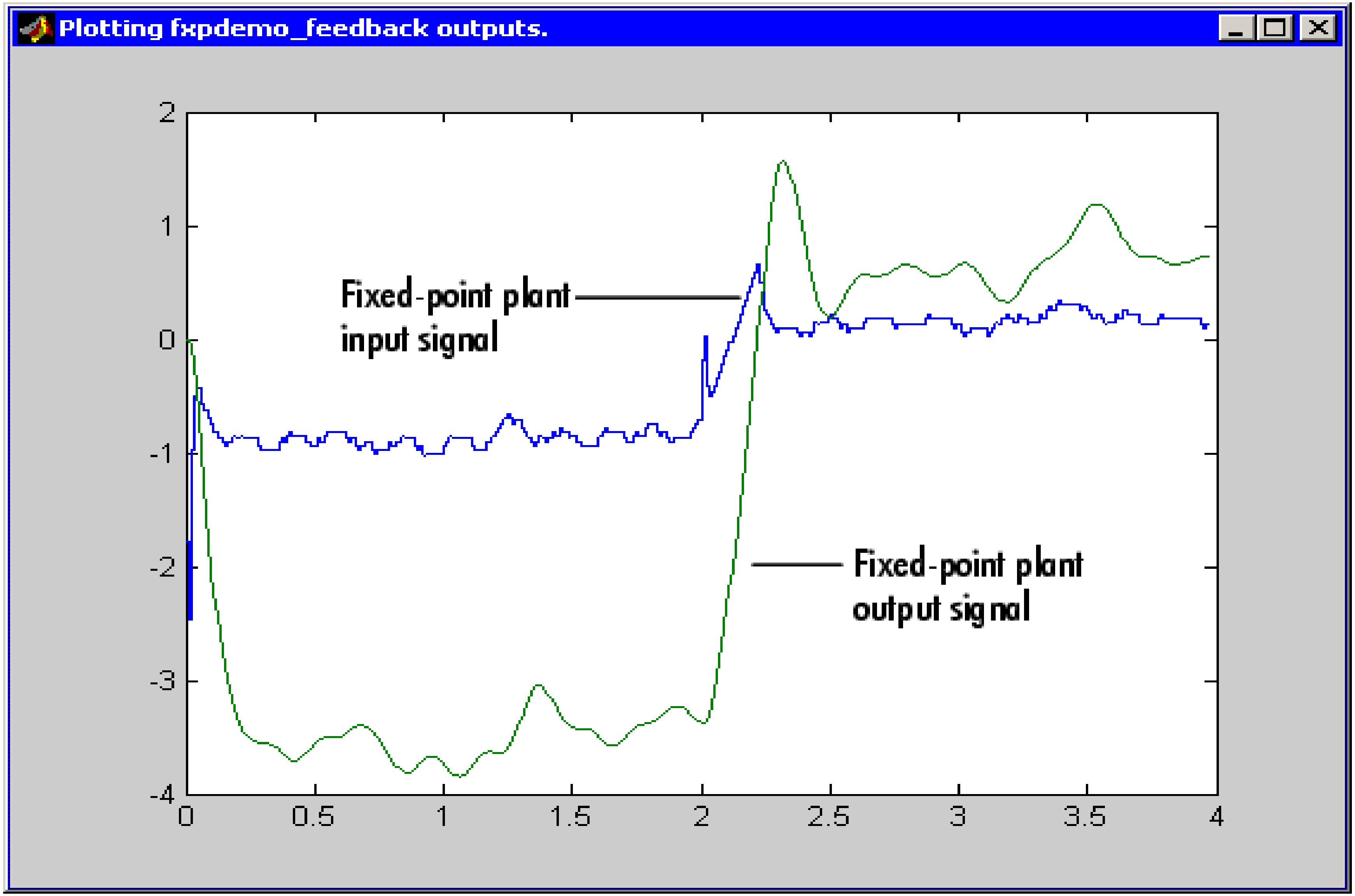
Feedback controller layout



Step input to the system

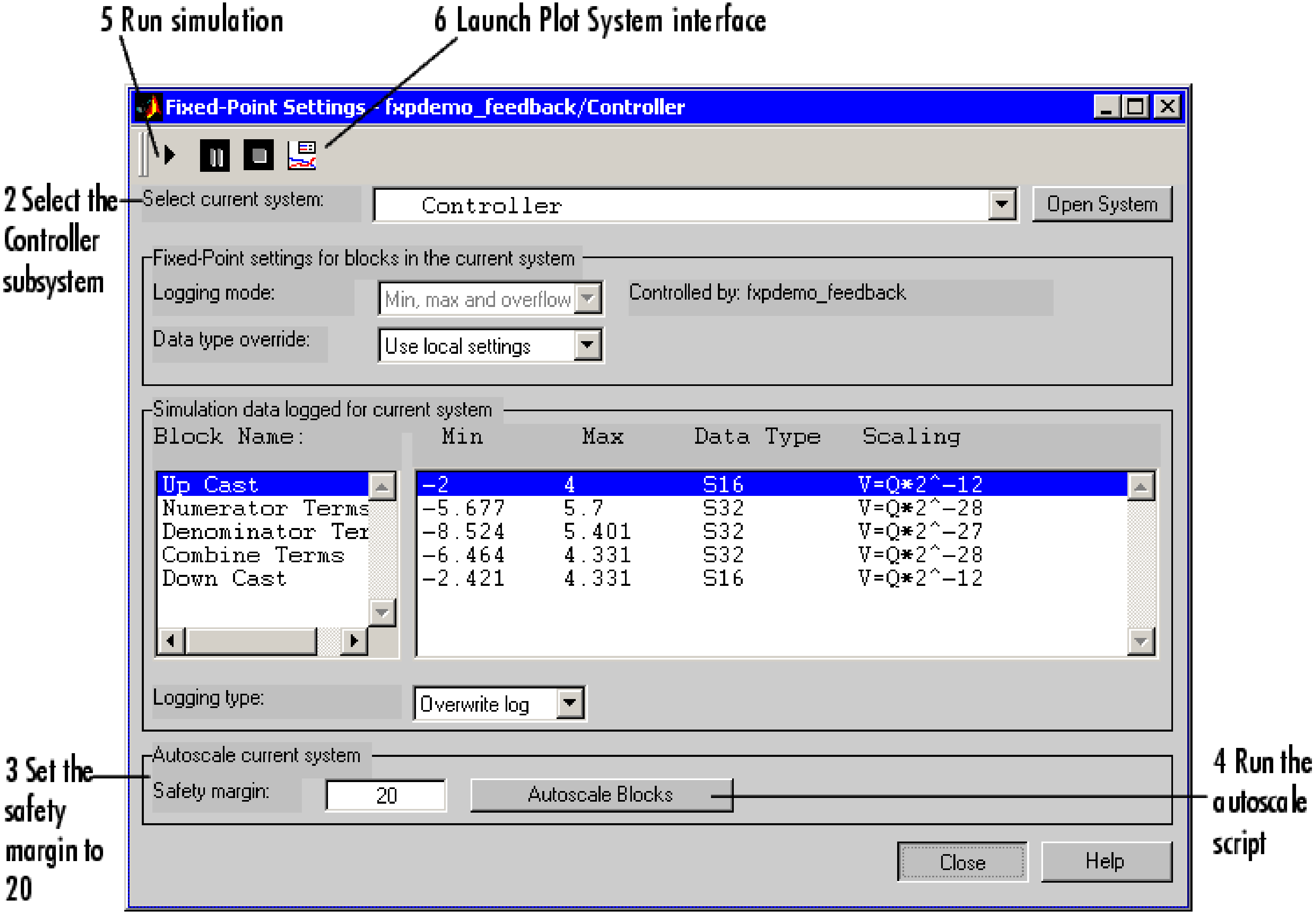


Unscaled results



Auto scaling

- The Fixed Point blockset aids in scaling all the variables in the model:



5 Run simulation

6 Launch Plot System interface

2 Select the Controller subsystem

3 Set the safety margin to 20

4 Run the autoscale script

Fixed-Point Settings - fxpdemo_feedback/Controller

Select current system: Controller

Fixed-Point settings for blocks in the current system

Logging mode: Min, max and overflow

Controlled by: fxpdemo_feedback

Data type override: Use local settings

Simulation data logged for current system

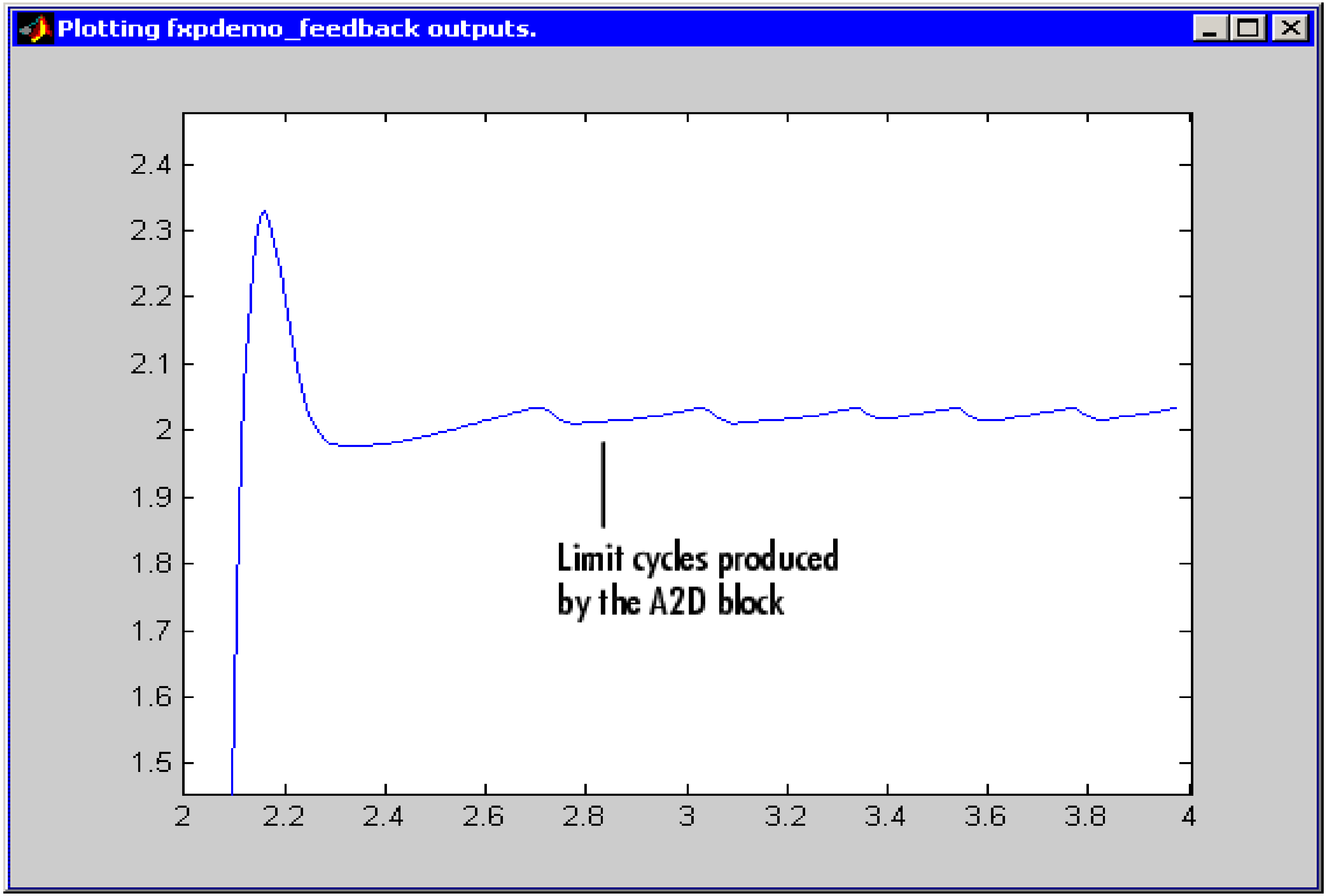
Block Name:	Min	Max	Data Type	Scaling
Up Cast	-2	4	S16	V=Q*2^-12
Numerator Terms	-5.677	5.7	S32	V=Q*2^-28
Denominator Ter	-8.524	5.401	S32	V=Q*2^-27
Combine Terms	-6.464	4.331	S32	V=Q*2^-28
Down Cast	-2.421	4.331	S16	V=Q*2^-12

Logging type: Overwrite log

Autoscale current system

Safety margin: 20

Output of the scaled model



Limit cycles produced by the A2D block

conduct the experiment yourself

- To work through this case yourself, type `fxpdemo_feedback` at the cmd prompt.
- Run the simulation without and with autoscaled integers.

End

- End of today's session and of the Matlab/Simulink course.

Good luck in all your engineering endeavours !