TELEMED Doppler RF data tools for the research in the MATLAB and Python environments

ArtUs Doppler Viewer

User Manual

Application version 1.0.0 · Manual version 1.0 · 09/11/2026

TELEMED
ULTRASOUND MEDICAL SYSTEMS

Internet page: http://www.pcultrasound.com

Information E-mail: info@pcultrasound.com

Support E-mail: support@pcultrasound.com

Contents

Introduction **3**

1. Doppler RF data recording and file structure **4**

2. Installation notes **6**

2.1. Running the standalone applications 7

3. Graphical user interface structure **9**

3.1. Main window and common controls 9

3.2. Color Doppler window (mode 1) 10

3.3. Pulsed Wave Doppler window (mode 2) 11

4. Signal processing algorithms **13**

4.1. Common pre-processing 13

4.2. Color flow (CFM) and B-mode reconstruction 13

4.3. Pulsed Wave Doppler spectrum, envelopes and audio 15

5. Functions reference **17**

6. Recommended sequence of actions and export formats **19**

7. Notes, limitations and troubleshooting **20**

8. References **21**

9. Revision History **22**

Introduction

ArtUs Doppler Viewer opens Doppler RF/IQ recordings made with the ArtUs RF Data Control sample and shows them as colour flow images, Doppler spectra and raw RF lines. It exists in two equivalent implementations with the same layout, controls, defaults and algorithms: a MATLAB App (`DopplerViewerApp.m`) and a Python/Qt application (`doppler_viewer.py`).

The tool works off-line on beamformed data and makes every processing step visible and adjustable. It is intended for research and teaching, not for diagnosis.

The application selects its view from the `mode` field of the file header:

Mode	Header name	View
0	BRF	B-mode image built from the RF window, with RF line review
1	BCD (Color Doppler)	B-mode + colour flow (velocity, power or variance), RF line review, cine playback
2	BPW (Pulsed Wave)	Range-gated Doppler spectrogram, velocity envelopes, stereo Doppler audio, RF line at the gate, Excel export

The screenshots show the MATLAB version; the Python version has the same panels and plot arrangement. Software screens may differ slightly from the figures, as the interface depends on the chosen options.

1. Doppler RF data recording and file structure

Recordings are made with the native C++ SDK sample ArtUs RF Data Control, which stores RF window data from the selected beamformer source point to a binary `*.bin` file. The file name holds the acquisition time, date and probe code (e.g. `07_27_36_09-11-2026_L15-7H40-A5.bin`).

For Doppler work, record in BCD (colour flow) or BPW (pulsed wave) scanning mode. Save from the Cine buffer after Freeze rather than while scanning.

RF data source points. The data are 16-bit integers taken from one point of the beamformer:

source_ID	Source point	Data in file	How the viewer uses it
1	Beamformer output	I (16 bit)	Hilbert transform builds I + jQ
2	TFC filter output	I (16 bit)	Hilbert transform builds I + jQ
3	Angle apodization output	I (16 bit)	Hilbert transform builds I + jQ
4	Hilbert transform output	I and Q planes (16 + 16 bit)	Used directly as I + jQ

Structure of one sub-frame (file type RF0005). The 6-byte tag `RF0005` is written once at the start of the file. Each sub-frame then contains:

Field	Data type	Number of elements	Description
number_of_frames	int32	1	Frames in the file (0 if recorded with Start/Stop)
header_size	int32	1	Header size in bytes (not used by the reader)
frame_size	int32	1	Frame size in bytes (not used by the reader)
mode	int32	1	0 = BRF, 1 = BCD, 2 = BPW; selects the viewer layout
source_ID	int32	1	RF data source point (table above)
tx_frequency	int32	1	Transmit frequency f0 [Hz]
frame_rate	int32	1	Frames per second × 100
Length_of_RF_row	int32	1	L: samples per RF line
Number_of_RF_rows	int32	1	R: lines (beams) in the RF window
Sampling_period_ns	int32	1	Ts: 25 ns (40 MHz) for ArtUs
sample_size	int32	1	Bits per sample (16 or 32)
start_depth	int32	1	Window start depth [mm]
DopplerPacketSize	int32	1	P: shots per beam (ensemble); 1 = no packet
Start_point_position_and_orientation	int32	3 × R	Triplets (beam_x [µm], beam_y [µm], angle [rad × 10^6])
time_stamps	uint32	R × P	Start time of every firing, in sampling periods

Field	Data type	Number of elements	Description
frame	int16	L × P × R (× 2 for source_ID 4)	RF samples: sample index fastest, then packet shot, then beam; for source_ID 4 the whole I plane is followed by the whole Q plane

The reader returns each frame as an array of L samples × R beams × P shots. Beam start coordinates are converted to cm and shifted by start_depth along the beam direction:

$$x_{beam} = \frac{beam_x}{10^4} + \frac{start_depth}{10}\sin\theta, \qquad y_{beam} = \frac{beam_y}{10^4} + \frac{start_depth}{10}\cos\theta$$

Older files use the tags RF0001–RF0004 (11 header fields, no `mode` or `DopplerPacketSize`; RF0003/RF0004 carry R time stamps) or no tag at all (7 header fields). Without a `mode` field the viewer opens a file as colour flow when P > 1, otherwise as B-mode.

2. Installation notes

Both versions run without extra toolboxes; choose the one that fits your environment.

MATLAB version.

Item	Requirement
MATLAB	R2020b or newer (uifigure, uigridlayout, uiaxes)
Toolboxes	None
Files on the path	`DopplerViewerApp.m`, `import_RFdata2MATLAB.m`
Start	`app = DopplerViewerApp` or `app = DopplerViewerApp('C:\data\file.bin')`

Python version.

Item	Requirement
Python	3.9 or newer
Required packages	numpy, matplotlib, PySide6
Optional packages	openpyxl (Excel export), sounddevice (audio; Windows/macOS wheels include PortAudio), scipy (export to .mat, faster wall filter)
Files in one folder	`doppler_viewer.py`, `artus_rf.py`, `requirements.txt`
Install	`pip install -r requirements.txt`
Start	`python doppler_viewer.py` or `python doppler_viewer.py C:\data\file.bin`

On Linux, audio also needs the PortAudio system library (e.g. `sudo apt install libportaudio2`). Without `sounddevice` everything except audio playback works; the Play button then shows a message.

The reader alone can be used from scripts:

```
RF, RFQ, H, probe = import_rf_data('file.bin')   # Python
[RF, RFQ, H, probe, flag] = import_RFdata2MATLAB(dir, name);   % MATLAB
```

2.1. Running the standalone applications

Both versions are also delivered as Windows executables for 64-bit Windows 10/11, so neither MATLAB nor Python has to be installed on the user's PC. Both executables have the same functions as the source versions, including audio and Excel export.

Item	MATLAB executable	Python executable
Delivered as	Installer (setup / `MyAppInstaller_web.exe`) or `ArtUsDopplerViewer.exe`	Folder `ArtUsDopplerViewer` with `ArtUsDopplerViewer.exe` and the `_internal` folder
Needed on the PC	Free MATLAB Runtime of the same release that was used to build the program (R2023a).	Nothing
Disk space	Small .exe plus the MATLAB Runtime (several GB, installed once)	About 150–250 MB
Start time	10–30 s on the first start; the splash screen is shown while the Runtime loads	About 2–3 s
Export result	`.mat` file (variable `doppler_result`)	`.mat` or `.npz` file

Installing the MATLAB executable.

1. Run the installer. It installs ArtUs Doppler Viewer and, if it is missing, downloads and installs the matching MATLAB Runtime. The web installer needs an internet connection, and installing the Runtime needs administrator rights.
2. Start the program from the Windows Start menu or from the installation folder.
3. Without the installer: install the MATLAB Runtime compatible with R2023a MATLAB (free download from www.mathworks.com/products/compiler/matlab-runtime.html), then run `ArtUsDopplerViewer.exe`.

Installing the Python executable.

1. Copy or unzip the complete folder `ArtUsDopplerViewer` to any location, e.g. `C:\Programs\ArtUsDopplerViewer`. Keep `ArtUsDopplerViewer.exe` and the `_internal` folder together in the same folder.
2. Run `ArtUsDopplerViewer.exe`. For a Desktop icon, right-click the .exe and choose Send to → Desktop (create shortcut); do not move the .exe itself out of its folder.
3. If Windows SmartScreen shows "Windows protected your PC", click More info → Run anyway; the program is not code-signed.

Opening a file. Use Load .bin file…, drag a `.bin` file onto the .exe or its shortcut, or start it from the command line: `ArtUsDopplerViewer.exe "D:\data\file.bin"`.

If the executable does not start:

Message or symptom	Cause and solution
“Failed to load Python DLL … python3xx.dll”	The `_internal` folder is missing or belongs to another build. Delete the program folder and copy the complete `ArtUsDopplerViewer` folder again.
MATLAB executable reports a missing MATLAB Runtime	The Runtime is not installed or is a different release. Install the release MATLAB Runtime R2023a.
Long wait with the splash screen	Normal for the first start of the MATLAB version while the Runtime loads.
Window opens and closes at once, or files disappear	The antivirus program may have quarantined files. Restore them and add the program folder as an exception.

3. Graphical user interface structure

The window has a scrollable control column on the left and two plot areas on the right. The control column always shows the Load button, the file information, the Acquisition panel and the Export button; below the Acquisition panel it shows either the Color Doppler panel (modes 0 and 1) or the Pulsed Wave panel (mode 2). The plot layout also follows the mode: RF line beside the image for colour flow, RF line above the spectrum for PW.

3.1. Main window and common controls

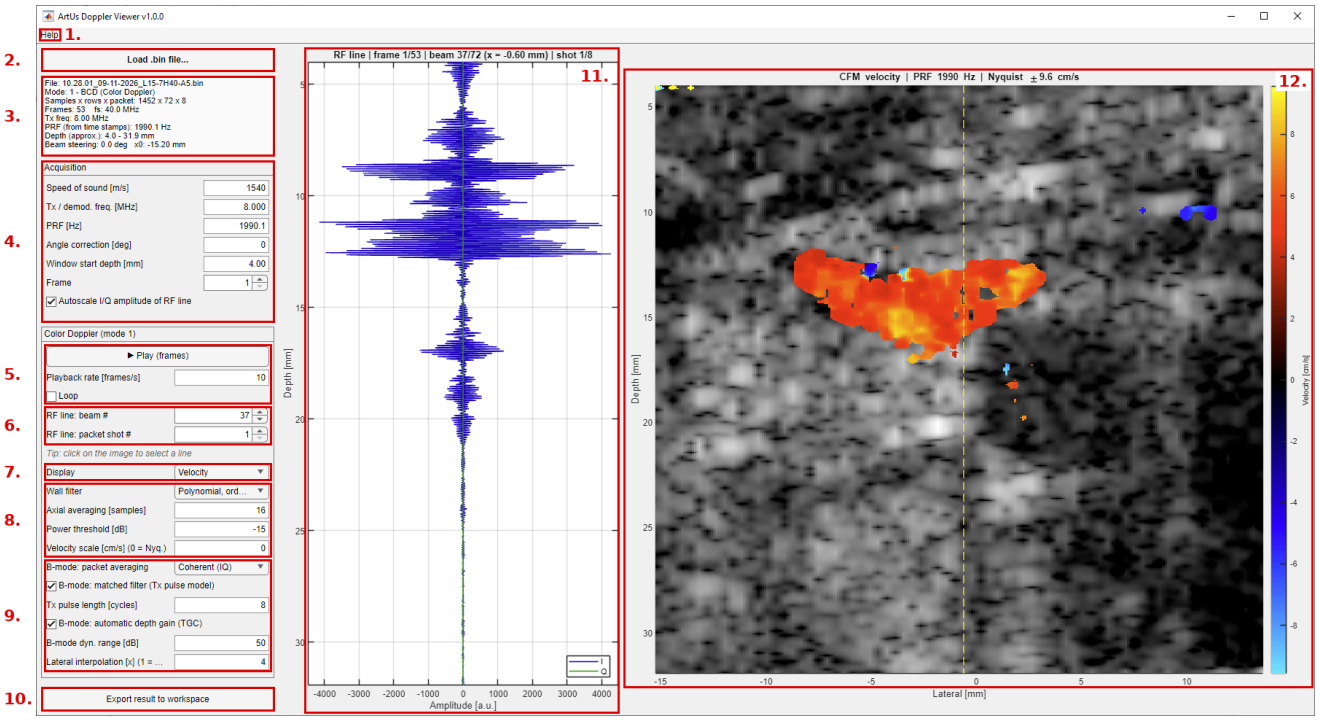


Fig. 1. Main window of the MATLAB version in Color Doppler mode (recording L15-7H40-A5, 8 MHz, PRF 1990 Hz, 72 beams × 8 shots, 53 frames). Components 1–4 and 10–11 are common to all modes; the Export result button (10) is at the bottom of the control column.

Num	Component	Description
1	Help menu	Help → About shows the application name, version (also shown in the window title, e.g. "ArtUs Doppler Viewer v1.0.0"), manufacturer and contact details.
2	Load .bin file	Opens a file dialog and imports an ArtUs RF data file. The layout, panels and default values are set from the file header.
3	File information	Read-only summary: file name, mode, samples × rows × packet (L × R × P), number of frames, sampling frequency, Tx frequency, PRF estimated from the time stamps, approximate depth range, beam steering angle and lateral start x0.
4	Acquisition panel	Parameters shared by all modes. **Speed of sound** [m/s] (default 1540) sets the depth axis and the velocity scale. **Tx / demod. freq.** [MHz] is read from the header and used for demodulation and velocity. **PRF** [Hz] is estimated from the time stamps and can be overwritten. **Angle correction** [deg] divides all velocities by cos θ. **Window start depth** [mm] is read from the header (start_depth plus beam start position)

Num	Component	Description
		and offsets every depth axis. **Frame** selects the recorded frame. **Autoscale I/Q amplitude of RF line** fits the amplitude axis of the RF line plot to the line shown; unticked, a fixed scale (frame or file maximum) is used.
10	Export result	Saves the last computed result with all parameters. MATLAB: workspace variable `doppler_result`. Python: `.mat` file (variable `doppler_result`, needs scipy) or `.npz` file.
11	RF line plot	Raw I (blue) and Q (green) samples of one RF line of the selected frame. For plain RF files Q is zero, because the Hilbert transform is applied only in processing. The title shows the frame, beam, packet shot or pulse time and lateral position.

All numeric fields accept a value on Enter or when focus leaves the field; out-of-range values are clipped to the limits. Every parameter change recomputes and redraws the view, and stops any running playback.

3.2. Color Doppler window (mode 1)

In modes 0 and 1 the RF line is drawn vertically against depth on the left (11), and the B-mode / colour flow image on the right (12). Numbers refer to Fig. 1.

Num	Component	Description
5	Cine playback	**Play (frames)** plays the recording frame by frame; the button becomes **Stop**. **Playback rate** [frames/s] (default 10) sets the speed; **Loop** restarts from frame 1. The selected beam and packet shot stay fixed and the RF amplitude axis is frozen at the largest value of that line over all frames. The B-mode and colour image are recomputed for every frame.
6	RF line selection	**RF line: beam #** (default: centre beam) and **packet shot #** select the line shown in the RF plot. A click on the image selects the nearest beam; the selected beam is marked by a dashed yellow line on the image.
7	Display	**Velocity** (default), **Power**, **Variance** or **B-mode only**.
8	Doppler processing	**Wall filter**: None, Mean removal, Polynomial order 1 or **Polynomial order 2** (default); the order is limited to P − 2. **Axial averaging** [samples] (default 16) averages the autocorrelation along depth. **Power threshold** [dB] (default −15) hides colour where the Doppler power is below this level relative to the frame maximum. **Velocity scale** [cm/s] fixes the colour range; 0 = ± Nyquist velocity.
9	B-mode reconstruction	**Packet averaging**: Coherent (IQ, default) or Incoherent (envelope). **Matched filter** (default on) with **Tx pulse length** [cycles] (default 8). **Automatic depth gain (TGC)** (default on). **B-mode dyn. range** [dB] (default 50). **Lateral interpolation** [×] (default 4; 1 = off).
12	Image	Grey-scale B-mode with colour overlay where the Doppler power exceeds the threshold. Velocity map: red–yellow = flow towards

Num	Component	Description
		the probe, blue–cyan = away. The title shows the display type, PRF and Nyquist velocity; the colour bar shows the scale.

3.3. Pulsed Wave Doppler window (mode 2)

In mode 2 the RF line at the gate is drawn above (9) and the Doppler spectrogram below (10). All frames are joined along slow time into one continuous pulse sequence.

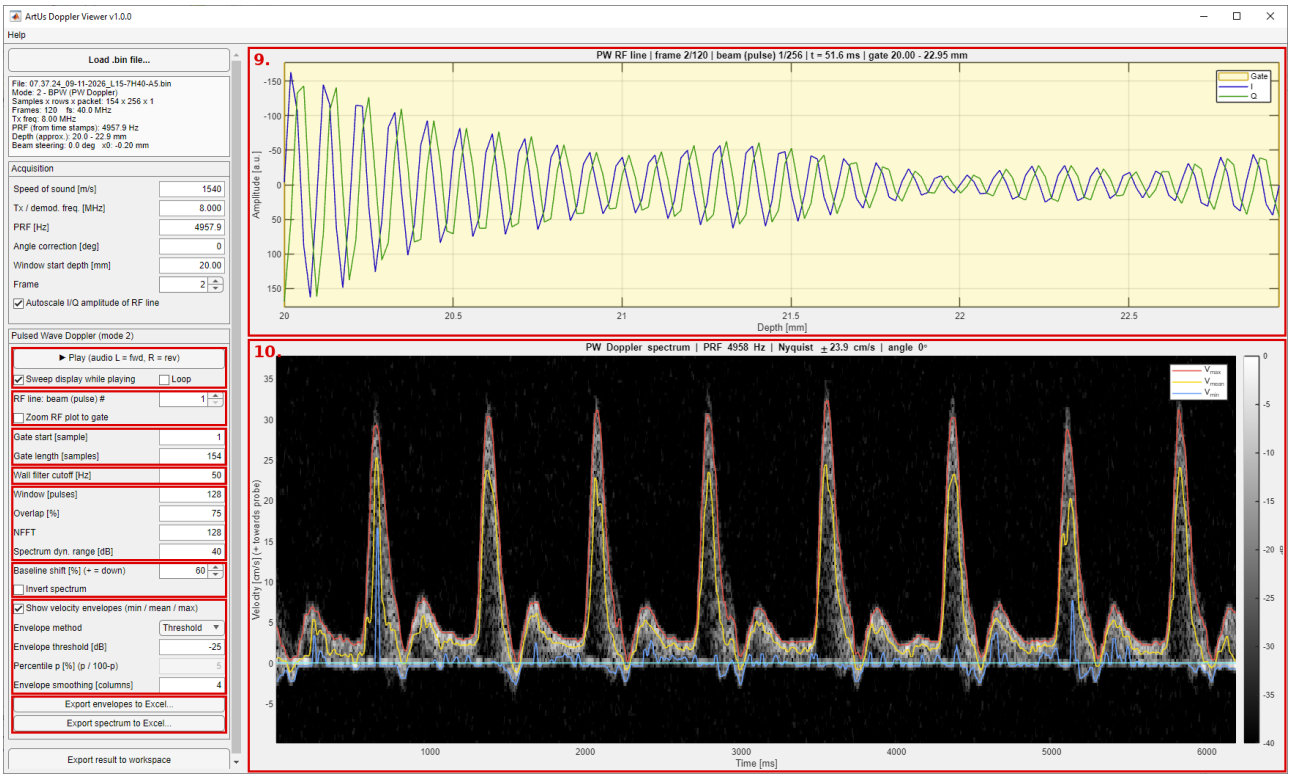


Fig. 2. Main window of the MATLAB version in Pulsed Wave mode (recording L15-7H40-A5, 8 MHz, PRF 4958 Hz, 120 frames, baseline shift 65 %). The Excel export buttons (8) are further down the panel.

Num	Component	Description
1	Audio playback	**Play (audio L = fwd, R = rev)** plays the Doppler signal as stereo sound: flow towards the probe on the left channel, away on the right. A red play-head runs over the spectrogram; with **Sweep display while playing** the not-yet-played part is black and the envelopes are revealed with it. The RF plot follows the pulse being played. **Loop** repeats; the button acts as **Stop** while playing.
2	RF line selection	**RF line: beam (pulse) #** selects the pulse of the current frame shown in the RF plot. **Zoom RF plot to gate** limits the depth axis to the gate ± 10 %.
3	Range gate	**Gate start** [sample] and **Gate length** [samples] define the sample volume; default = the whole RF line. The gate is shaded yellow in the RF plot, and its depth in mm is shown in the plot title.
4	Wall filter	**Wall filter cutoff** [Hz] (default 50) of the first-order high-pass filter that removes slow tissue and wall motion; 0 = off.

Num	Component	Description
5	Spectrogram	**Window** [pulses] (default 128), **Overlap** [%] (default 75), **NFFT** (default 128; never smaller than the window) and **Spectrum dyn. range** [dB] (default 40).
6	Display shift	**Baseline shift** [%] moves the zero-velocity line (+ = down, step 10 %), so aliased high velocities can be shown unwrapped. **Invert spectrum** swaps the sign: + then means away from the probe.
7	Velocity envelopes	**Show velocity envelopes** draws Vmax (red), Vmean (yellow) and Vmin (blue). **Envelope method**: Threshold or Percentile. **Envelope threshold** [dB] (default −25) re. the spectrogram maximum. **Percentile p** [%] (default 5; Percentile method only). **Envelope smoothing** [columns] = running median length (1 = off).
8	Excel export	**Export envelopes to Excel** and **Export spectrum to Excel** write `.xlsx` files (section 6).
9	RF line plot	I and Q of the selected pulse versus depth, with the gate shaded. With Autoscale on, the amplitude axis fits the visible part of the line.
10	Doppler spectrogram	Grey-scale spectrum in dB versus time [ms] and velocity [cm/s], with the zero line in cyan. The title shows the PRF, Nyquist velocity and angle.

4. Signal processing algorithms

Both implementations use identical algorithms; function names are listed in section 5. Notation: n = fast-time sample (depth), p = packet shot, k = pulse (slow time), Ts = sampling period, f0 = Tx frequency, c = speed of sound, θ = angle correction.

4.1. Common pre-processing

Analytic signal. Files with an I and a Q plane (source_ID 4) are used directly. For plain RF the Q component is obtained by a Hilbert transform along depth, computed with an FFT (positive frequencies doubled, negative set to zero):

$$z(n) = I(n) + jQ(n), \qquad Q = \mathcal{H}\{I\} \tag{1}$$

The mean along depth is then subtracted from every line to remove DC offset.

Depth axis. The depth of sample n starts at the window start depth z0:

$$d(n) = z_0 + n\,\frac{c\,T_s}{2}\cos\theta_{steer} \tag{2}$$

θsteer is the beam steering angle from the header (used in PW mode; 0 for linear-array CFM).

PRF from time stamps. Time stamps count sampling periods. With P > 1 the PRF uses the intervals between consecutive shots of the same beam; with P = 1, the intervals between consecutive lines:

$$PRF = \frac{1}{\mathrm{median}(\Delta t_s)\,T_s} \tag{3}$$

Only positive intervals are used, so counter wrap-around and frame gaps do not bias the estimate.

4.2. Color flow (CFM) and B-mode reconstruction

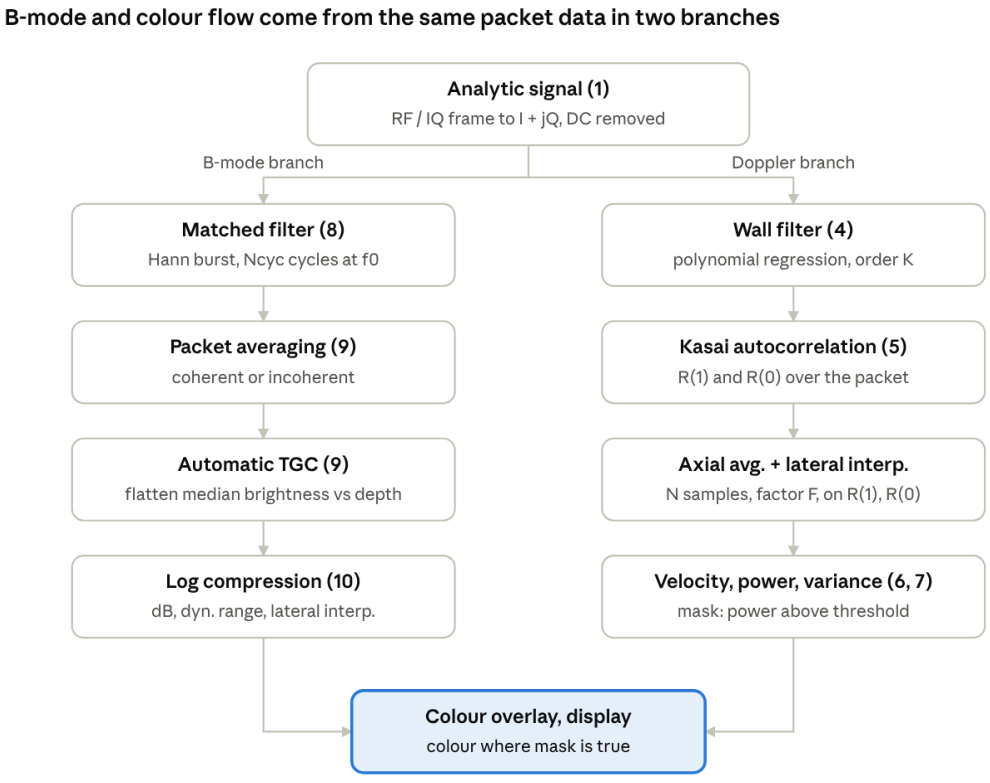


Fig. 3. Colour flow and B-mode processing chain (equation numbers in brackets)

The B-mode branch (left) and the Doppler branch (right) both start from the same analytic packet data; colour replaces grey only where the Doppler power passes the threshold.

Wall (clutter) filter. Polynomial regression along the packet removes the components that vary slowly over the P shots (stationary and slowly moving tissue). With t = p / (P − 1) and Q the orthonormal basis (QR decomposition) of the polynomials 1, t, …, t^K:

$$z_f = z - Q\,Q^T z, \qquad K = \min(\text{order},\, P-2) \tag{4}$$

"Mean removal" is K = 0 and "None" skips the filter.

Kasai autocorrelation estimator. Lag-one and lag-zero autocorrelations over the packet:

$$R(1) = \frac{1}{P-1}\sum_{p=1}^{P-1} z_f^*(p)\, z_f(p+1), \qquad R(0) = \frac{1}{P}\sum_{p=1}^{P} \left|z_f(p)\right|^2 \tag{5}$$

Both are averaged along depth with a moving mean of length N (Axial averaging), then interpolated laterally (factor F). Interpolating R(1) rather than the velocity avoids false colours where neighbouring beams alias.

Velocity, power, variance:

$$f_D = \frac{PRF}{2\pi}\arg R(1), \qquad v = \frac{c\, f_D}{2 f_0 \cos\theta}, \qquad v_{Nyq} = \frac{c\, PRF}{4 f_0 |\cos\theta|} \tag{6}$$

$$P_{dB} = 10\log_{10}\frac{R(0)}{\max R(0)}, \qquad \sigma^2 = 1 - \frac{|R(1)|}{R(0)} \in [0,1] \tag{7}$$

Positive velocity means flow towards the probe. Colour is drawn only where P_dB exceeds the Power threshold.

B-mode from the Doppler packets. Three optional steps precede log compression:

1. Matched filter: each line is correlated with a model of the transmit pulse, a Hann-windowed burst of Ncyc cycles at f0 (Np = round(Ncyc / (f0 Ts)) samples, unit energy). The output is shifted by (Np − 1)/2 samples so echoes keep their depth.
2. Packet averaging: coherent (complex mean, lowers the noise floor by about 10·log10 P dB; fast blood partly cancels) or incoherent (mean of magnitudes).
3. Automatic TGC: the lateral median brightness per depth, smoothed over L/20 samples, is flattened.

$$y(n) = \sum_{m=0}^{N_p-1} z\,(n+m)\, p^*(m), \qquad p(m) = w_{Hann}(m)\, e^{\,j2\pi f_0 T_s m} \tag{8}$$

$$env_{coh} = \left|\frac{1}{P}\sum_p y_p\right|, \qquad env_{inc} = \frac{1}{P}\sum_p \left|y_p\right|, \qquad env_{TGC} = \frac{env}{g(d)}\,\text{median}(g) \tag{9}$$

$$B = 20\log_{10}\frac{env}{\max env}, \qquad grey = \text{clip}\left(\frac{B+DR}{DR},\, 0,\, 1\right) \tag{10}$$

B is interpolated laterally with the same factor as the Doppler data. The colour overlay replaces the grey value wherever the power mask is true; the colour index is round((value − low) / (high − low) × 255) into a 256-entry map.

4.3. Pulsed Wave Doppler spectrum, envelopes and audio

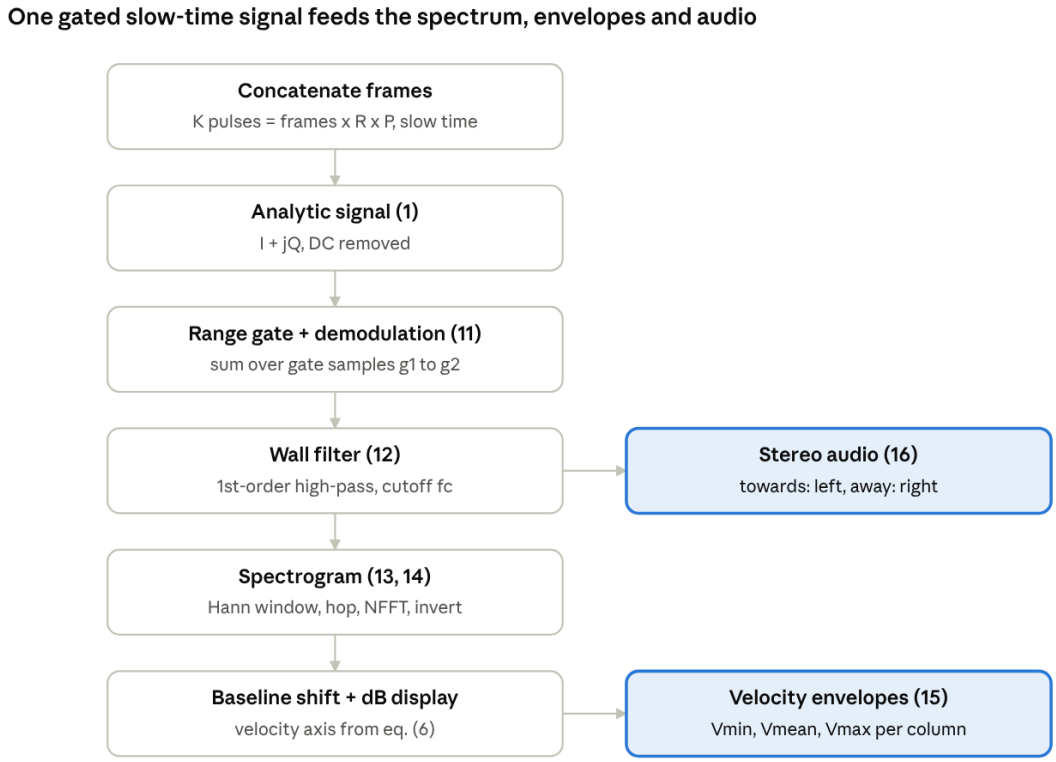


Fig. 4. Pulsed Wave Doppler processing chain (equation numbers in brackets)

The audio is taken from the wall-filtered signal and the envelopes from the displayed spectrum, so both follow every setting above them.

Slow-time signal. All frames are concatenated into one sequence of K pulses (R × P per frame). Inside the gate [g1, g2] each pulse is demodulated at f0 and summed (range-gate integration); the mean is removed:

$$s(k) = \sum_{n=g_1}^{g_2} z\,(n,k)\, e^{-j2\pi f_0 T_s (n-1)}, \qquad s \leftarrow s - \bar{s} \tag{11}$$

Wall filter. First-order high-pass with cutoff fc:

$$y(k) = s(k) - s(k-1) + a\, y(k-1), \qquad a = e^{-2\pi f_c / PRF} \tag{12}$$

Spectrogram. Short-time Fourier transform with a Hann window of Nw pulses, hop H = round(Nw (1 − overlap)), zero-padded to NFFT (at least Nw), and FFT-shifted so zero Doppler is in the middle:

$$S(f_i, m) = \sum_{k=0}^{N_w-1} w\,(k)\, s(k+mH)\, e^{-j2\pi ik/N_{FFT}}, \qquad w(k) = 0.5 - 0.5\cos\frac{2\pi k}{N_w - 1} \tag{13}$$

$$f_i = i\,\frac{PRF}{N_{FFT}}, \quad i = -N_{FFT}/2\,, \ldots, N_{FFT}/2 - 1, \qquad t_m = \frac{mH + N_w/2}{PRF} \tag{14}$$

Invert uses s* instead of s. Baseline shift b [%] rotates the periodic spectrum by sb = round(b/100 × NFFT/2) bins and moves the frequency axis by sb × PRF / NFFT. The display is S_dB = 20 log10(|S| / max |S|) over the spectrum dynamic range, and frequency maps to velocity as in (6).

Velocity envelopes. Per spectrogram column, only bins above the envelope threshold T count, weighted by their power:

$$M_i = [S_{dB}(f_i) > T], \qquad W_i = |S(f_i)|^2 M_i, \qquad v_{mean} = \frac{\sum_i v_i W_i}{\sum_i W_i} \tag{15}$$

- Threshold method: Vmin and Vmax are the lowest and highest velocity bins with M = 1.
- Percentile method: with the cumulative power C_i = Σ_{l ≤ i} W_l / Σ W, Vmin is the first bin where C ≥ p and Vmax the first where C ≥ 1 − p. This is more robust to isolated noise bins.
- Columns with no bin above T give NaN (a gap in the curve, an empty Excel cell).
- Optional smoothing: running median over N columns, ignoring NaN; gaps stay gaps.

Envelopes are taken from the displayed spectrum, so they already include angle correction, baseline shift and inversion.

Stereo audio. The complex signal is split into its positive and negative Doppler halves with an FFT, and each half is played as one channel:

$$X = \mathrm{FFT}(s), \quad left = \mathrm{Re}\,\mathrm{IFFT}(X\,[1 \le k < N/2]), \quad right = \mathrm{Re}\,\mathrm{IFFT}(X\,[k > N/2]) \qquad (16)$$

The two channels are scaled to 90 % of full scale. One pulse is one audio sample: MATLAB plays at round(PRF) Hz, and Python resamples (FFT method) to 44.1 kHz, which keeps the timing identical. The play-head time is the number of samples played divided by the sample rate, so the spectrogram cursor, the sweep and the RF line stay synchronised with the sound.

5. Functions reference

The processing functions are local functions in `DopplerViewerApp.m` and module-level functions in `doppler_viewer.py`, so they can be reused in scripts. Arrays are L samples × R beams × P shots unless stated.

No.	MATLAB	Python	Inputs	Output	Purpose
1	`import_RFdata2MATLAB`	`import_rf_data` (artus_rf.py)	directory + file name / path	RF, RFQ (cell / list per frame), HEADER, probe code (+ flag)	Read an ArtUs `.bin` file (section 1)
2	`getComplex`	`get_complex`	I, Q arrays	complex z, DC removed	Analytic signal, eq. (1)
3	`analyticSig`	`analytic_signal`	real RF	complex analytic signal	Hilbert transform via FFT
4	`estimatePRF`	`estimate_prf`	header, R, P, Ts	PRF [Hz] or NaN	Eq. (3)
5	`drawCFM` (method)	`draw_cfm` (method)	parameters	image, result struct	Wall filter, Kasai, eq. (4)–(7), display
6	`bmodeEnvelope`	`bmode_envelope`	z, parameters, Ts	envelope L × R	Matched filter, packet averaging, TGC, eq. (8)–(9)
7	`matchedFilterFastTime`	`matched_filter_fast_time`	z, f0, Ts, cycles	filtered z, same size	Eq. (8)
8	`autoTGC`	`auto_tgc`	envelope	gain-corrected envelope	Automatic depth gain
9	`latInterp`	`lat_interp`	A (L × R), x, factor F	interpolated A, x	Linear lateral interpolation in beam-index space
10	`velocityMap`	`velocity_map`	n colours	n × 3 colour map	Blue–black–red velocity map
11	`pwSignal` (method)	`pw_signal` (method)	parameters	slow-time s, gate, depth	Eq. (11)–(12)
12	`drawPW` (method)	`draw_pw` (method)	parameters	spectrogram, result struct	STFT, baseline shift, eq. (13)–(14)
13	`spectralEnvelopes`	`spectral_envelopes`	S_dB, power, velocity axis, threshold, method, p, smoothing	Vmin, Vmean, Vmax per column	Eq. (15)
14	`smoothKeepGaps`	`movmedian_nan`	vector, length	smoothed vector	Running median that keeps NaN gaps
15	`onPlay` (method)	`on_play` (method)	–	stereo audio	Eq. (16), playback and play-head

No.	MATLAB	Python	Inputs	Output	Purpose
16	`onExportEnvelopes`, `onExportSpectrum`	`on_export_envelopes`, `on_export_spectrum`	current result	`.xlsx` file	Excel export (section 6)

6. Recommended sequence of actions and export formats

Color Doppler (mode 1).

1. Load the `.bin` file (Load .bin file…) and check the file information: mode 1, packet size P > 1, plausible PRF.
2. Check Speed of sound, Tx frequency and Window start depth; set Angle correction if the vessel angle is known.
3. Adjust the B-mode first: Tx pulse length close to the real Doppler pulse, Packet averaging, TGC and dynamic range.
4. Set the Wall filter (order 2 for P ≥ 4) and the Power threshold so that colour fills the vessel without noise in tissue; raise Axial averaging for smoother colour.
5. Click on the image to choose a beam and inspect its RF line and packet shots.
6. Press Play (frames) to review the sequence; use Loop and Playback rate as needed.
7. Export result to save the velocity, power and variance maps.

Pulsed Wave Doppler (mode 2).

1. Load the file and check the PRF estimated from the time stamps.
2. Place the gate on the vessel: set Gate start and Gate length, then use Zoom RF plot to gate to confirm its position.
3. Set the Wall filter cutoff just above the tissue motion (typically 50–150 Hz).
4. Choose Window and NFFT (default 128 / 128) and the spectrum dynamic range; use Baseline shift or Invert if the flow aliases or is displayed on the wrong side.
5. Set the envelope threshold until Vmax follows the spectral edge; switch to Percentile or add smoothing (3–5 columns) if Vmax is spiky.
6. Press Play to listen and watch the sweep; the RF plot follows the pulse being played.
7. Export the envelopes and/or the spectrum to Excel.

Excel files. Both files include an **Info** sheet with all processing parameters (file, PRF, f0, c, angle, Nyquist velocity, gate, wall cutoff, window, overlap, NFFT, baseline, velocity sign, envelope method, threshold, smoothing, export time).

File	Sheet	Content
`<name>_envelopes.xlsx`	Envelopes	Time_ms, Vmin / Vmean / Vmax [cm/s], Fmin / Fmean / Fmax [Hz]; empty cell = no signal above threshold
`<name>_spectrum.xlsx`	Spectrum_dB	Rows = time [ms] (column A), columns = velocity [cm/s] (row 1); values in dB re. the spectrogram maximum
`<name>_spectrum.xlsx`	Axis	Bin number, velocity [cm/s] and Doppler frequency [Hz] of every column

A spectrum larger than one Excel sheet (16384 columns × 1048576 rows) is refused with a message; reduce NFFT or the overlap.

Result export. The result structure (`doppler_result`) holds the axes, the B-mode image in dB and, depending on the mode, the velocity / power / variance maps and mask, or the slow-time signal, spectrogram, envelopes and gate, plus the processing parameters and the file name.

7. Notes, limitations and troubleshooting

Situation	Explanation / action
PRF shows NaN or a wrong value	No usable time stamps (old file types) or gaps in the recording; type the correct PRF in the Acquisition panel, since all velocities scale with it.
Flow appears on the wrong side	The sign depends on the IQ convention of the source point; use Invert spectrum (PW) or note the convention for CFM.
Envelope data are not phase-coherent	Detected or log-compressed data carry no phase, so no Doppler can be computed; record RF or IQ (source_ID 1–4).
IQ data already at baseband	The viewer assumes the carrier is still at f0 and demodulates again; Doppler stays correct, but long gates lose SNR through partial cancellation.
Matched filter changes little	ArtUs Doppler echoes are already narrow-band (e.g. −6 dB band 7.7–8.3 MHz at 8 MHz) and out-of-band noise is removed in the receiver; coherent packet averaging gives the larger noise gain. Pulse polarity (sign) has no effect on any displayed quantity.
Blocky B-mode or colour image	Few beams in the colour window; raise Lateral interpolation. Interpolation smooths the display but adds no lateral resolution.
First pulse of a PW file is flat	The device may record a zero first line; it is real data, not a reader error.
Spiky Vmax envelope	Raise the envelope threshold, use the Percentile method or smoothing of 3–5 columns.
No sound (Python)	Install `sounddevice` (and PortAudio on Linux); all other functions work without it.
Excel export refused	The spectrum exceeds one sheet; reduce NFFT or the overlap.

The tools are for research and teaching on recorded data. They are not a medical device and must not be used for diagnosis.

8. References

1. C. Kasai, K. Namekawa, A. Koyano, R. Omoto. Real-time two-dimensional blood flow imaging using an autocorrelation technique. IEEE Transactions on Sonics and Ultrasonics, vol. 32, no. 3, pp. 458–464, 1985.
2. S. Bjærum, H. Torp, K. Kristoffersen. Clutter filter design for ultrasound color flow imaging. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 49, no. 2, pp. 204–216, 2002.
3. D. H. Evans, W. N. McDicken. Doppler Ultrasound: Physics, Instrumentation and Signal Processing, 2nd ed. Wiley, 2000.
4. J. A. Jensen. Estimation of Blood Velocities Using Ultrasound: A Signal Processing Approach. Cambridge University Press, 1996.
5. TELEMED. RF data tools for the research in the MATLAB environment, User Manual, Ver. 1.15, 11/04/2022.
6. Echo Blaster, LS, ClarUs, SmartUs, MicrUs and ArtUs Series Ultrasound Systems, Ultrasonography for Windows II Software Development Kit (Usgfw2 SDK) Programmer's Guide.

9. Revision History

Revision	Revision Date	Description of Revision	Revision Author
1.0	2026.11.09	Initial release	A.Sakalauskas