| Supporting Information

The following paragraphs will give a brief overview of the submissions with respect to the architecture and toolboxes used. Major differences between the submissions, especially implementation-wise, will be highlighted.

University of California, Berkeley

Based on Python this submission uses an interactive Jupyter notebook to run the reconstruction. The CG algorithm is implemented using the SigPy python package [32], which provides a high level abstraction interface to run code on CPUs or GPUs. Coil sensitivity profiles for the head scans are estimated via a root-sum-of-squares (SoS) approach, dividing each channel by the SoS reconstruction. The sensitivity profiles for the heart data are estimated using the BART toolbox [16]. Density compensation is done by taking the norm of each coordinate in the trajectory, resulting in a simple ramp function. The NUFFT algorithm is based on Fessler's min-max interpolation [14]. Data is multiplied with the square root of the density compensation to preserve adjointness of the density compensated forward and backward operator (given in equation 3) in the CG algorithm. The CG algorithm is terminated after $\{3, 6, 15, 50\}$ iterations. No k-space filtering is performed after the CG reconstruction. Brain and heart data are undersampled by skipping acquired lines according to the desired undersampling factor.

Berlin Ultrahigh Field Facility (B.U.F.F)

This Matlab based submission uses the NUFFT from the BART toolbox in the self-written CG implementation. Coil sensitivity estimation is performed via a SoS approach, dividing each channel by the SoS reconstruction. Density correction is performed by taking the Euclidean norm of each trajectory position in 2D, normalized by the maximum value. Intensity correction is achieved by normalizing the sum over the complex coil sensitivity to yield one. The CG algorithm is run on the brain data for 150 iterations for no acceleration and for 100 iterations for the accelerated cases. For the heart data set 40 iterations are used for all cases. The filter function for k-space position k_x, k_y is given in equation 6 and k_c being half the cropped FOV size, $\beta = 100$ are used as parameters. The filter is applied to the cropped FOV image after the CG algorithm has terminated.

Eindhoven University of Technology

This submission uses Python to achieve the desired reconstruction results. The main package used for reconstruction is PyNUFFT [37] which also supplies the CG algorithm. The internal NUFFT algorithm is based on Fessler's min-max interpolation [14]. Each coil channel is reconstructed independently using PyNUFFT and the final image is formed via a SoS approach. No density correction or coil sensitivity estimation is performed prior to reconstruction. The CG algorithm is terminated after 11 iterations. No k-space filtering is performed after reconstruction. Brain and heart data are undersampled by skipping acquired lines according to the desired undersampling factor.

Swiss Federal Institute of Technology Zurich (ETH)

This implementation is built on code which was presented at past educational sessions and image reconstruction workshops of the ESMRMB and provides an easy to comprehend, Matlab based implementation of the CG-SENSE algorithm. The sensitivity maps are calculated from the fully-sampled center of the radial k-space data using an SVD-based approach by Walsh et al. [18]. The gridding operator is devised as a sparse matrix with a Kaiser-Bessel kernel function to utilize the fast matrix-vector multiplications in Matlab. A noise covariance matrix of the receive channels can be included in the reconstruction if available. The iteration is not regularized (e.g. by Tikhonov regularization), but stopped manually after a number of iterations determined by visual assessment of the intermediate result. In the last step, a k-space filter is applied to suppress noise from k-space areas with no acquired data.

Karolinska Institutet (KI)

This submission is written in Matlab, using the object oriented capabilities of the language. The aim is to make the code as readable as possible. Coil sensitivities are estimated by dividing each coil element by the SoS of all coil elements. The CG algorithm is implemented from scratch using the same variable names and formalism as the original paper [1], to help readers relate the code to the paper. A “SENSE-operator” is implemented, which performs the steps outlined in the algorithm description provided in the original paper. A linear ramp-filter is used for sampling density compensation, and the gridding and Fourier transform is performed using the NUFFT routine, with Kaiser-Bessel interpolation, from the Michigan Image Reconstruction Toolbox (MIRT) [14]. No additional filtering of the k-space data is performed.

New York University (NYU)

This submission is a straightforward implementation of the method described in [1] using available tools in Matlab. The implementation builds on the NUFFT toolbox by Jeff Fessler [14] for the radial reconstruction and on the Matlab portion of the gpuNUFFT package by Andreas Schwarzl and Florian Knoll [38] for the conjugate gradient algorithm implementation [1]. The theoretical density compensation function (k/max(k)) was used and coil sensitivities were estimated by applying an adaptive reconstruction [18] of single coil images. The CG algorithm is run for 100 iterations and augmented using Tikhonov regularization with a regularization weight of 0.2. If the residual values are below $1e^{-6}$ the algorithm is terminated. No k-space filtering is applied after the reconstruction.

Stanford University

The submission is based on Python and implements gridding via a Cython module. Gridding is realized via linear interpolation of a pre-computed Kaiser-Bessel kernel function. The visualization of the reconstruction steps is done in an interactive Jupyter Notebook. Density compensation is realized via a simple ramp function computed from the

Euclidean norm of the 2D k-space grid positions. The input data is multiplied with a Hamming window with the parameter M amounting to the number of samples on each spoke. Coil estimation is performed via a SoS approach, by filtering the data with a narrow-width Gaussian kernel ($\sigma = 5\%$ k-space width), applying the inverse NUFFT to the filtered data and dividing each channel by the SoS reconstruction. In contrast to the other submissions, FOV cropping is performed to center the brain in the reconstructed image instead of cropping symmetrical around the center. The CG algorithm is run for 8 iterations for brain and heart data. The filter function for the normalized k-space position k_x, k_y is given in equation 6 and $k_c = 1$, $\beta = 100$ are used as parameters. The filter is applied in every iteration of the CG algorithm.

Graz, University of Technology, Institute of Computer Graphics and Vision (TUG H.)

The submission is based on Python and uses the gpuNUFFT [38], which is written in C++/CUDA to accelerate the reconstruction operator. Wrapper scripts to run the gpuNUFFT in Python are provided. The gridding in the gpuNUFFT is realized via nearest neighbour interpolation of a pre-computed Kaiser-Bessel function, where the gridding kernel size is set to 3. Coil sensitivity maps are estimated using ESPIRiT [33] from the BART toolbox [16]. Intensity correction is applied by dividing the coil sensitivity maps by its SoS reconstruction, resulting in unit-norm along the coil channels. Density compensation is applied using a ramp filter estimated from the provided trajectories. To achieve adjointness of the forward and adjoint reconstruction operator E and E^H, the raw data is multiplied by the square-root of the density compensation function. For reconstruction, 100 iterations are performed in the CG algorithm which was extended with Tikhonov regularization ($\lambda = 0.2$).

Graz, University of Technology, Institute of Medical Engineering (TUG M.)

The submission is based on Python and makes use of OpenCL to accelerate the reconstruction process. The Code is packaged and supports installation with automated dependency detection. Coil sensitivities are estimated from all acquired spokes using the ESPIRiT algorithm [33] implemented by the BART toolbox. Intensity correction is achieved by normalizing the sum over the complex coil sensitivity to yield one. As density compensation, a simple ramp function is used. The NUFFT is based on a Kaiser-Bessel gridding approach with linear interpolation between the pre-computed kernel points. The data is multiplied by the square root of the density compensation to ensure adjointness of the forward and backward application of the NUFFT. The CG algorithm is extended by an Tikhonov regularization and run for 300 iterations. The regularization weight λ is chosen as 0.5. The filter function for k-space position k_x, k_y is given in equation 6 and $k_c = 25$, $\beta = 100$ are used as parameters. The filter is applied to the cropped FOV image after the CG algorithm has terminated.

University of Southern California (USC)

This Matlab implementation uses in-house written C/Mex function to implement the reconstruction algorithm. Gridding is realized via sinc interpolation combined with the Matlab FFT implementation. As density compensation, a simple ramp function is used. Coil sensitivity estimation is performed via a SoS approach using a 32-by-32 low resolution k-space center region and dividing each channel by the SoS reconstruction. Intensity correction is achieved by normalizing the sum over the complex coil sensitivity to yield one. The CG algorithm is run for $\{3, 6, 15, 50\}$ iterations for the brain data set and for 6 iterations for the heart data set. The filter function for the normalized k-space position k_x, k_y is given in equation 6 and $k_c = 0.5, \beta = 100$ are used as parameters. The filter is applied to the cropped FOV image after the CG algorithm has terminated.

Utah Center for Advanced Imaging Research, University of Utah (Utah)

This Matlab based submission uses a CPU/GPU accelerated NUFFT implementation, based on min-max interpolation of Fessler and Sutton [14]. Coil sensitivities are estimated by the method of Walsh et al. [18]. Sensitivity maps are normalized to give a sum of one along the coil dimension, accounting for intensity variations and ensuring adjointness. Gridding is performed via linear interpolation of a pre-computed Kaiser-Bessel kernel. Density compensation amounts to a simple ramp. Reconstruction is run for $\{200, 100, 100, 100\}$ iterations for brain and 100 iterations for heart data. The filter function for k-space position k_x, k_y is given in equation 6. k_c is chosen to be half of the cropped FOV and $\beta = 100$ is used as parameter. The filter is applied to the cropped FOV image after the CG algorithm has terminated.

Massachusetts General Hospital (MGH)

This submission demonstrates how to use Matlab and modify the BART toolbox to perform a CG-SENSE reconstruction. Coil sensitivities are estimated using the ESPIRiT algorithm [33] of BART. In addition to the forward/backward application of the NUFFT, a faster Toeplitz-embedding based implementation is described. Reconstruction is performed by a modification of BART's "pics" method, the BART implementation of parallel imaging and compressed sensing reconstruction. Performance is demonstrated using all acquired projections of the brain data set only. No scripts to produce the desired images from undersampled data or for the heart data were submitted.

SUPPORTING INFORMATION TABLE S1 Sequence and k-space related parameters of the supplied data sets.

	Challenge data		Supplementary Data	
	Brain	Heart	Cardiac	Brain
Type	radial spin-echo	radial FLASH	radial bSSFP	spiral
TR in ms	2500	2.22	3.14	2000
TE in ms	50	1.32	1.57	25
Flip angle in °	n/a	10	50	90
Matrix - acquisition	256x256	160x160	256x256	220x220
Matrix - reconstruction	300x300	240x240	154x154	240x240
Projections/Interleaves	96	5x11	420	3
Readoutpoints	512	320	256	27121
FOV acquisition in mm^2	200x200	256x256	358x358	220x220
Slice thickness in mm	2	6	8	2
Recieve Coil	12	34	18 + 12(8)	16
Main field in T	3	3	1.5	3