FUNM_QUAD: AN IMPLEMENTATION OF A STABLE QUADRATURE-BASED RESTARTED ARNOLDI METHOD FOR MATRIX FUNCTIONS

ANDREAS FROMMER*, STEFAN GÜTTEL†, AND MARCEL SCHWEITZER*

This note gives an overview of the FUNM_QUAD MATLAB code which implements the restarted Arnoldi algorithm described in [4] and analysed in [3]. Parts of FUNM_QUAD have been adopted from the FUNM_KRYL code [1], and FUNM_QUAD also implements deflated restarting based on the analysis in [2].

FUNM_QUAD can be downloaded from either one of the following web sites:

http://www.guettel.com/funm_quad

http://www-ai.math.uni-wuppertal.de/SciComp/software/funm_quad.html

The code can be used to approximate $f(A)\boldsymbol{b}$, the action of a matrix function on a vector, for an arbitrary (Hermitian or non-Hermitian) matrix A, a vector $\boldsymbol{b}$, and a function f with an integral representation

$$f(z) = \int_\Gamma \frac{g(t)}{t-z}\,\mathrm{d}t. \tag{1}$$

For details concerning the algorithm we refer the reader to [4].

The basic calling sequence of FUNM_QUAD is

```
[f,out] = funm_quad(A,b,param),
```

where `A` is a (sparse) quadratic matrix, `b` is a vector of corresponding length, and `param` controls various parameters (including the function f) of the algorithm. The output parameter `f` corresponds to the final approximation to $f(A)\boldsymbol{b}$, while the structure `out` collects various other outputs. In the following we describe the possible input and output parameters in detail.

Inputs:

- `param.function` (string or function handle): The function f to be evaluated. Predefined functions are `'invSqrt'` for $f(z) = z^{-\frac{1}{2}}$, `'exp'` for $f(z) = e^z$, and `'log'` for $f(z) = \log(1+z)/z$. Other functions can be evaluated by specifying a function handle for the integrand in (1).
- `param.restart_length` (integer): The number of Arnoldi steps performed in each restart cycle.
- `param.max_restarts` (integer): The maximum number of restart cycles to be performed.
- `param.tol` (scalar): The error tolerance for numerical quadrature.
- `param.hermitian` (0 or 1): Specifies whether `A` is Hermitian.
- `param.V_full` (0 or 1): Specifies whether the full Arnoldi basis should be stored and returned in the `out` structure.

*Department of Mathematics, Bergische Universität Wuppertal, 42097 Wuppertal, Germany, {frommer,schweitzer}@math.uni-wuppertal.de

†School of Mathematics, The University of Manchester, M13 9PL Manchester, United Kingdom, stefan.guettel@manchester.ac.uk

- `param.H_full` (0 or 1): Specifies whether all Hessenberg matrices should be stored and returned in the `out` structure.
- `param.exact` (vector or `[]`): If the exact solution $f(A)\boldsymbol{b}$ is known it can be passed to `FUNM_QUAD` for computation of the error after each cycle.
- `param.stopping_accuracy` (scalar): Relative accuracy at which the algorithm is terminated.
- `param.inner_product` (function handle): The inner product used for orthogonalization.
- `param.thick` (function handle or `[]`): Thick-restart function for implicitly deflated restarts. Typically, this will be the function `thick_quad` provided with our code.
- `param.number_thick` (integer): Number of target eigenvalues to be deflated when thick restarts are used.
- `param.min_decay` (scalar between 0 and 1): Desired rate of linear error reduction. If this rate is no longer achieved, the algorithm terminates.
- `param.reorth_number` (0 or 1): Number of reorthogonalizations in Arnoldi's method.
- `param.truncation_length` (integer of `inf`): Truncation length for Arnoldi's method.
- `param.transformation_parameter`: Parameter used in the integral transformation when dealing with $f(z) = z^{-\frac{1}{2}}$. For details on the choice of this parameter, see [4].
- `param.waitbar` (0 or 1): Specifies whether a waitbar indicating the progress of the algorithm is shown.
- `param.verbose` (0 or 1 or 2): The level of information outputted on the command line while running the algorithm.

Outputs:

- `out.stop_condition`: Specifies why the algorithm terminated (maximum number of iterations reached, achieved desired accuracy etc.).
- `out.V_full`: Full Arnoldi basis (if desired).
- `out.H_full`: Hessenberg matrices from all restart cycles (if desired).
- `out.time`: CPU time needed for each restart cycle.
- `out.thick_interpol`: Interpolation nodes (Ritz values) from each restart cycle.
- `out.thick_replaced`: Additional interpolation nodes from thick restart procedure for each cycle (if used).
- `out.num_quadpoints`: Number of quadrature points used for evaluating the error function in each restart cycle.
- `out.appr`: Arnoldi approximation after each restart cycle.
- `out.update`: Update of the Arnoldi iterate after each restart cycle.
- `out.err`: Euclidean norm of the error after each restart cycle (if exact solution is provided as input).

For more details and examples on how to use `FUNM_QUAD`, see also the different demo files `demo_*.m` provided with our code. When using `FUNM_QUAD` or referring to it, please consider citing the paper [4].

REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 229–314.

[2] M. Eiermann, O. G. Ernst, and S. Güttel, *Deflated restarting for matrix functions*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 621–641.

[3] A. Frommer, S. Güttel, and M. Schweitzer, *Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 661–683.

[4] ———, *Efficient and stable Arnoldi restarts for matrix functions based on quadrature*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1602–1624.