Documentation for the kernelsolver routine in MATLAB

Syntax

```
[K,KB] = kernelsolver(L,M,S,W,T,Q,N)
[K,KB] = kernelsolver(L,M,S,W,T,Q,N,Ny)
[K,KB] = kernelsolver(L,M,S,W,T,Q,N,Ny,msg,qe,cfs)
[K,KB,rd,ef] = kernelsolver(...)
```

Description

The routine solves continuum kernel equations of the form

$$\mu(x)\partial_x k(x,\xi,y) - \lambda(\xi,y)\partial_\xi k(x,\xi,y) - k(x,\xi,y)\partial_\xi\lambda(\xi,y) = \theta(\xi,y)\bar{k}(x,\xi) + \int_0^1 \sigma(\xi,\eta,y)k(x,\xi,\eta)d\eta, \quad (1a)$$

$$\mu(x)\partial_x\bar{k}(x,\xi) + \mu(\xi)\partial_\xi\bar{k}(x,\xi) + \mu'(\xi)\bar{k}(x,\xi) = \int_0^1 W(\xi,y)k(x,\xi,y)dy, \quad (1b)$$

where $\mu, \lambda, \sigma, \theta, W$, and q are given parameters and $(k, \bar{k})$ is the sought solution, i.e., the continuum kernels. The equations (1) are defined on a prismatic domain $0 \le \xi \le x \le 1, y \in [0,1]$ and equipped with boundary conditions

$$k(x,x,y) = -\frac{\theta(x,y)}{\lambda(x,y)+\mu(x)}, \quad (2a)$$

$$\mu(0)\bar{k}(x,0) = \int_0^1 q(y)\lambda(0,y)k(x,0,y)dy, \quad (2b)$$

for all $x \in [0,1]$ and for almost every $y \in [0,1]$. The basis of the routine is to compute solutions to (1), (2) as power series approximations of the form

$$k(x,\xi,y) \approx \sum_{\ell=0}^{N}\sum_{i=0}^{N-\ell}\sum_{j=0}^{i} K_{ij\ell}x^{i-j}\xi^j y^\ell, \quad (3a)$$

$$\bar{k}(x,\xi) \approx \sum_{i=0}^{N}\sum_{j=0}^{i} \bar{K}_{ij}x^{i-j}\xi^j, \quad (3b)$$

where the approximation order N is determined by the user.

As an alternative to the generic approximation order N, the routine has an option to use a reduced approximation order N_y in y, which reduces the number of unknown coefficients from $\mathcal{O}(N^3)$ to $\mathcal{O}(N_y N^2)$. In this case, the power series approximation (3a) becomes

$$k(x,\xi,y) \approx \sum_{\ell=0}^{N_y}\sum_{i=0}^{N-\ell}\sum_{j=0}^{i} K_{ij\ell}x^{i-j}\xi^j y^\ell. \quad (4)$$

The routine additionally provides the option to use the exact function q in the computations instead of its Taylor series approximation $q(y) \approx \sum_{i=0}^{N} q_i y^i$, which may reduce approximation errors. However, this requires that the integral in (2b) can be computed analytically, when λ and k are replaced by their power series approximations[1]. For more details about this and the power series approach in general, see [1, Sect. 3].

The routine additionally has the optional subroutine `closedform` to check if the exact solution to (1), (2) can be found in closed-form, in which case the (sub)routine returns the exact, closed-form solution. The subroutine is based on [1, Sect. 4], i.e., it checks if the parameters of (1), (2) satisfy the necessary assumptions for the closed-form solution to exist. In more detail, this involves checking that

- parameters λ and μ are constant;
- parameters σ, W, and θ are separable, i.e., they can be written as

$$W(x,y) = W_x(x)W_y(y), \tag{5a}$$
$$\sigma(x,y,\eta) = \sigma_x(x)\sigma_y(y)\sigma_\eta(\eta), \tag{5b}$$
$$\theta(x,y) = \theta_x(x)\theta_y(y), \tag{5c}$$

  for some functions, $W_x, W_y, \sigma_x, \sigma, y, \sigma_\eta, \theta_x, \theta_y$;
- There exist a constant c_y such that

$$c_y = \frac{\sigma_\eta(y)}{\theta_y(y)} \int_0^1 \sigma_y(\eta)\theta_y(\eta)d\eta, \tag{6}$$

  is satisfied independently of y, and

$$c_y\sigma_x'(\xi) + \lambda\frac{\theta_x''(\xi)\theta_x(\xi) - \theta_x'(\xi)^2}{\theta_x(\xi)^2}W_x(\xi)\theta_x(\xi)\int_0^1 W_y(y)\theta_y(y)dy, \tag{7}$$

  holds for all $\xi \in [0,1]$.

If any of the checks fail, the subroutine terminates, providing a reason for termination, and if all the checks are successful, the subroutine returns the exact, closed-form solution as[2]

$$k(x,\xi,y) = -\frac{1}{\lambda+\mu}\exp\left(\frac{c_x}{\mu}x\right)\exp\left(-\frac{c_x}{\mu}\xi\right)\theta_x(\xi)\theta_y(y), \tag{8a}$$
$$\bar{k}(x,\xi) = \exp\left(\frac{c_x}{\mu}x\right)\bar{k}_\xi(\xi), \tag{8b}$$

where

$$\bar{k}_\xi(\xi) = \left(\frac{c_y}{\lambda+\mu}\sigma_x(\xi) - \frac{c_x}{\mu} + \frac{\lambda}{\lambda+\mu}\frac{\theta_x'(\xi)}{\theta_x(\xi)}\right)\exp\left(-\frac{c_x}{\mu}\xi\right), \tag{9}$$

and

$$c_x = \frac{\mu}{\lambda+\mu}\left(c_y\sigma_x(0) + \lambda\frac{\theta_x'(0)}{\theta_x(0)} + \frac{\lambda}{\mu}\theta_x(0)\int_0^1 q(y)\theta_y(y)dy\right). \tag{10}$$

[1] Employing integration by parts, this reduces to being able to integrate q over $[0,1]$ analytically.

[2] The solution provided by the subroutine may not be exactly (8)–(10) due to the finite accuracy of floating point computations, i.e., roundup errors of the order of the machine epsilon may appear.

`[K,KB] = kernelsolver(L,M,S,W,T,Q,N)` returns the approximate solution (3) (as symbolic expression of `x`, `z` (treated as ξ) and `y`) of order `N`, which must be a positive integer, to (1), (2). The other input parameters correspond to the parameter functions of (1), (2) as follows:

- `L` The parameter λ; must be a symbolic expression of `x` and `y`.
- `M` The parameter μ; must be a symbolic expression of `x`.
- `S` The parameter σ; must be a symbolic expression of `x`, `y`, and `h`.
- `W` The parameter W; must be a symbolic expression of `x` and `y`.
- `T` The parameter θ; must be a symbolic expression of `x` and `y`.
- `Q` The parameter q; must be a symbolic expression of `y`.

`[K,KB] = kernelsolver(L,M,S,W,T,Q,N,Ny)` returns the approximate solution (4), (3b) of reduced order `Ny` in y to (1), (2). The input `Ny` must be a positive integer not larger than `N`.

`[K,KB] = kernelsolver(L,M,S,W,T,Q,N,Ny,msg,qe,cfs)` solves the preceding problem using options `msg`, `qe`, and `cfs`, which are treated as logical values. Setting `msg` to true/false enables/disables display messages as the routine is running (default: true). Setting `qe` to true/false enables/disables the routine to use the exact q in the computations (default: false). Setting `cfs` to true/false enables/disables the subroutine `closedform` seeking the exact, closed-form solution (default: false).

`[K,KB,rd,ef] = kernelsolver(...)` solves the preceding problem with additional output arguments `rd` and `ef`. The output `rd` contains the error of the least squares fit after solving the set of linear equations for the coefficients $K_{ij\ell}$ and $\bar{K}_{ij}$. If the exact, closed-form solution is found, then `rd=0`. The output `ef` contains the exit flag of the `closedform` subroutine. The case `ef=0` indicates success (explicit, closed-form solution was found) and the other cases indicate failure due to the following reasons:

- `ef=1` The parameter λ is not constant.
- `ef=2` The parameter μ is not constant.
- `ef=3` The parameter σ is not separable.
- `ef=4` The parameter W is not separable.
- `ef=5` The parameter θ is not separable.
- `ef=6` There is no constant c_y satisfying (6).
- `ef=7` The condition (7) is not satisfied.
- `ef=8` The `closedform` subroutine was disabled.

Examples

See the `solverdemo.m` script for examples of using the routine.

References

[1] J.-P. Humaloja and N. Bekiaris-Liberis. On computation of approximate solutions to large-scale backstepping kernel equations via continuum approximation. arXiv, 2406.13612, 2024.