The MATEDOR project develops software technologies and standard APIs, along with a sustainable and portable library, for large-scale computations that can be broken down into very small matrix or tensor computations. The main target of MATEDOR is to accelerate applications from important fields that fit this profile, including deep learning, data mining, astrophysics, image and signal processing, hydrodynamics, and more.

**Sustainable and Performance-Portable Software**

MATEDOR is a high-performance numerical library for batched linear algebra subroutines autotuned for modern processor architectures and system designs. The MATEDOR library includes LAPACK-compliant routines that target many small dense problems, tensor, and application-specific operations, e.g., for deep-learning. These routines are constructed as much as possible out of calls to batch BLAS routines and their look-alikes required in sparse computation context.

**Standard Interface for Batch Routines**

Working closely with interested application communities, we define modular and language-agnostic interfaces that can be implemented to work seamlessly with compilers and to be optimized using techniques such as code generation and inlining. This enables application developers, compilers, and runtime systems to express computational workloads as one or more calls to standard batch routines, and would allow the entire linear algebra (LA) community to collectively attack a wide range of small matrix or tensor problems. Success in such an effort requires innovations in interface design, computational and numerical optimization, as well as packaging and deployment on the user side to trigger final stages of tuning at the moment of execution.

**Standard C Interface:**

1. Simple to the MAGMA library [3][4] and cuBLAS.
2. Pointer-to-pointer (P2P) interface. Problems are not necessarily equidistant from each other.
3. Separate APIs for fixed and variable size batch problems.

```c
void dgemm_batched(    // BLS batched calls to MAGMA
    char transA, char transB,      // transposes of A and B
    int m, int n, int k,        // A, B and C dimensions
    double alpha, double beta,    // scalar multipliers
    double const * A, double const * B, // A and B matrices
    double const * C,            // C is overwritten
    int lda, int ldb, int ldc,     // strides
    queue_t queue );            // this can be NULL

void dgemm_vbatched(   // BLS batched calls to MAGMA
    char transA, char transB,      // transposes of A and B
    int m, int n, int k,        // A, B and C dimensions
    double alpha, double beta,    // scalar multipliers
    double const * A, double const * B, // A and B matrices
    int batch, double * C,      // C is overwritten
    int lda, int ldb, int ldc,     // strides
    queue_t queue );          // this can be NULL
```

---

**Applications**

1. **Tensor Contractions in High-order FEM and Applications [1]**

2. **Dense/sparse System Solvers & Preconditioners [3][4]**

3. **Density Matrix Renormalization Group (DMRG++)**

4. **Distributed hierarchical linear solver [6]**

5. **Deep Neural Networks and Data Analytics [1]**

---

**References**