USER-DEFINED MATERIAL MODELING (UMAT) IN LS-DYNA

General UMAT Development Framework

UMAT Development Overview

Objective: Implement custom hyperelastic material models in LS-DYNA through user-defined subroutines

Key Components:

- `umatxx.f` subroutine interface
- Deformation gradient processing
- Stress computation framework
- History variable management

Implementation Stages:

1. Environment setup
2. Code compilation
3. Static linking
4. Validation testing

Critical Requirements

- Intel Fortran Compiler (ifx)
- LS-DYNA object files (LS-DYNA object-oriented package)
- Proper library linking (LS-DYNA, MKL, FFTW)

Step 1: Obtaining LS-DYNA Package

Download Requirements: The first step is to download the object-oriented package of LS-DYNA from LSTC Essential Files in Package :

- `dyn21usermats.f` - Main file to edit for UMAT implementation
- `dyn21usermat.f` - Material interface definitions where the components of deformation gradient and other history variables in the global and local frames are defined passing into `dyn21usermats.f`
- `dyn21usld.f` - Solid element routines
- `dyn21utan.f` - Tangent stiffness (for implicit analysis)
- `dyn21.f` - Main FORTRAN program file
- Object files (`.obj`) - Pre-compiled LS-DYNA routines
- `Makefile` - Build configuration file and custom executable

Note

`dyn21usermats.f` contains subroutine place holders or samples umat41-umat50, where each corresponds to a user-defined material ID

Step 2: Environment Setup - Required Software Installation

1. **Microsoft Visual Studio 2022**
   - Desktop development with C++ → Compiler toolchain
   - Windows SDK components → System libraries
2. **Intel oneAPI Base Toolkit 2025**
   - Intel Math Kernel Library (MKL) → Optimized matrix/tensor operations
   - FFTW (FFTW3) Interface to MKL → Fast Fourier transforms for spectral methods
   - Optimized BLAS/LAPACK routines → Linear algebra computations
   - Threading Building Blocks → Parallel processing support
3. **Intel oneAPI HPC Toolkit**
   - Intel Fortran Compiler 2025.0.4 (ifx)
   - Intel MPI Library (optional)
4. **Directory Setup**
   - Unpack Object-Oriented LS-DYNA contents into LS-DYNA installation directory

Important

Install Visual Studio before Intel oneAPI to ensure proper integration. The FFTW interfaces in MKL are essential for optimized math operations.

Step 3: Creating UMAT Subroutine - Edit `dyn21usermats.f`

1. Choose a UMAT slot (`umat41-umat50`) - we use umat45 here
2. Modify the selected subroutine to implement your material model
3. Key implementation aspects:
   - Extract material parameters from `cm(*)` array - values defined in input file keyword `*MAT_USER_DEFINED_MATERIAL_MODELS`
   - Retrieve deformation gradient from history variables `hsv(1:9)`
   - Use `Z(3,3)` for material coordinate system - first two columns define material frame directions
   - Compute stress response based on constitutive model (e.g., second Piola-Kirchhoff stress)
   - Return Cauchy stress in `sig(1:6)` array (Voigt notation)

Reference

See `*MAT_USER_DEFINED_MATERIAL_MODELS` in LS-DYNA Keyword User's Manual Volume II

Note: Default UMATs `umat41-umat50` are either placeholders or sample material model for shell elements; full replacement is required for hyperelastic solid elements

Step 4: Compilation Process

Compile Modified UMAT: Open Intel oneAPI Command Prompt for VS 2022 and navigate to source directory

```
ifx /c /nologo /fpp /O2 /MD /Qopenmp ^
    -DPCWIN -DINTEL -DAMD64 -DWINX64 ^
    -DLSTCODE -DLONGFORMAT ^
    dyn21usermats.f /Fo:dyn21usermats.obj
```

Performance Flags:

- `/c` - Compile only
- `/O2` - Level 2 optimization
- `/MD` - Multi-threaded runtime
- `/Qopenmp` - OpenMP support
- `/fpp` - Fortran preprocessor

Platform Definitions:

- `-DPCWIN` - Windows platform
- `-DINTEL` - Intel architecture
- `-DAMD64` - 64-bit support
- `-DWINX64` - Windows x64
- `-DLSTCODE` - LS-DYNA specific
- `-DLONGFORMAT` - Extended precision

Output: Creates `dyn21usermats.obj` file

Step 5: Environment Variables Setup

Configure Library Paths: Set environment variables for Intel MKL and FFTW libraries

```
set MKL64=%MKLROOT%\lib\intel64
set FFTW_WRAP=%USERPROFILE%\mkl_fftw3xf
```

Required Files:

- Ensure all `.F` Fortran source files are present
- Place `userinterface.mod` in compilation folder
- Verify all object files (`.obj`) associated with Fortran files listed in Step 1 are available, if any object files is missing or changed, follow the Compilation Process in Step 4 for the Fortran file.

Library Components:

- Intel MKL: `mkl_intel_lp64.lib`, `mkl_sequential.lib`, `mkl_core.lib`
- FFTW wrapper: `fftw3xf_ifx.lib`
- LS-DYNA: `libdyna.lib`, `libansys.lib`
- System: `user32.lib`, `advapi32.lib`

Step 6: Static Linking

Create Custom Executable: Link all object files and libraries to generate a custom LS-DYNA executable.

- Linking executable:

```
ifx /nologo /MD ^
  dyn21umats.obj ^
  dyn21cnt.obj dyn21tumat.obj dyn21ueos.obj dyn21umat.obj ^
  dyn21umatc.obj dyn21umatv.obj ^
  dyn21ushl.obj dyn21usld.obj dyn21utan_aniso.obj ^
  init_dyn21.obj dynrfn_user.obj dyn21.obj couple2other_user.obj ^
  dyn21em.obj dyn21extumat.obj dyn21icfd.obj invert3x3.obj ^
  /link /OUT:lsdyna_MSCM_dp.exe /STACK:16777216 /INCREMENTAL:NO /FORCE:MULTIPLE ^
  /LIBPATH:"%MKL64%" /LIBPATH:"%FFTW_WRAP%" ^
  fftw3xf_ifx.lib mkl_intel_lp64.lib mkl_sequential.lib mkl_core.lib ^
  libdyna.lib libansys.lib user32.lib advapi32.lib ^
  /NODEFAULTLIB:libifcoremt.lib /NODEFAULTLIB:libifportmt.lib /NODEFAULTLIB:LIBCMT
```

Output: `lsdyna_MSCM_dp.exe` with embedded UMAT

- Run the generated `lsdyna_MSCM_dp.exe` with your input file.

Step 7: Alternative Build Method

Using Makefile with `nmake`: Alternative to manual compilation and linking

1. Ensure `Makefile` is in the same folder as source files
2. Modify `Makefile` to include:
    - All necessary `.obj` files
    - LS-DYNA libraries paths
    - Intel Fortran compiler libraries
3. Open Intel oneAPI Command Prompt
4. Run: `nmake`

Note

The Makefile method automates the compilation and linking process but may require customization for your specific setup

Output: Same as manual method - `lsdyna_custom.exe`

Performance Optimization Results

Achieved Performance Benefits: Custom build demonstrates significant improvements over standard implementation

Intel MKL Integration:

- Optimized matrix operations
- Fast 33 tensor inversions
- Vectorized math functions
- BLAS/LAPACK routines

Runtime Optimization:

- Conflict-free library loading
- Multi-threaded execution
- CPU-optimized instructions
- Fast exponential evaluations

Performance Metrics

- 3-4 speedup compared to standard user material implementation
- Improved numerical stability through optimized libraries
- Reduced memory footprint with static linking
- Better cache utilization with compiler optimizations

Summary: UMAT Development Workflow

1. **Obtain LS-DYNA Package**
   - Download object-oriented SMP library from LSTC
2. **Setup Environment**
   - Install Visual Studio 2022 and Intel oneAPI
3. **Modify UMAT**
   - Edit `dyn21usermats.f` with custom material model
4. **Compile**
   - Use Intel Fortran compiler with optimization flags
5. **Link**
   - Create executable with MKL and LS-DYNA libraries
6. **Execute**
   - Run `lsdyna_custom.exe` with input files

Next Steps

With the framework established, we implement isotropic and anisotropic incompressible hyperelstic material models.

Transition to Specific Material Model Implementation

From Framework to Specific Models: Now that the general UMAT development framework is established, we proceed to implement specific hyperelastic material models

Part II: Neo-Hookean Model

- Isotropic nearly-incompressible hyperelastic material
- Simplest nonlinear model
- Baseline for validation
- 2 material parameters

Part III: GOH Model

- Anisotropic nearly-incompressible hyperelastic
- Fiber-reinforced material
- Dispersion effects
- 7 material parameters

Implementation Focus

Each model will detail:

- Mathematical formulation and strain energy function
- UMAT interface specifics (`cm(*)`, `hsv(*)` arrays and `Z(*,*)` usage)
- Fortran implementation with code snippets

Modified Structural Constitutive Model (MSCM)

Implementation

From Basic Models to Advanced Fiber Mechanics: Implementation of the MSCM for soft biological tissues with progressive fiber recruitment

Model Features

- Neo-Hookean matrix
- Progressive fiber recruitment
- Angular fiber distribution (ODF)
- Gaussian recruitment distribution
- 8 material parameters

Key Capabilities

- Captures nonlinear stiffening
- Models fiber dispersion (σ_n)
- Recruitment bounds (λ_{lb}, λ_{ub})
- FEFR/TFR tracking

Implementation Components

MSCM-specific implementation includes:

- Fiber integration with Gauss quadrature over segmented θ and λ_s
- Orientation distribution functions $\Gamma_0(\theta)$ and $\Gamma_t(\beta)$
- History variable: include quantities required for post-processing