TMATROM3 User Guide

M. Ganesh   Stuart C. Hawkins

1 Introduction

The TMATROM3 package implements the numerically stable T-matrix formulation described and numerically demonstrated in the accompanying paper [2]. The latest version of the package can be downloaded from

```
https://github.com/stuart-hawkins/tmatrom3/
```

We request that any publications making use of TMATROM3 cite [2].

In this user guide we briefly describe usage of the key functions and methods in the TMATROM3 package.

The process for simulating scattering of an incident wave using the T-matrix is visualised in Figure 1 and the associated functions and methods are documented in Sections 3–5. The process for obtaining the T-matrix is visualised in Figure 2 and the associated functions and methods are documented in in Sections 6–7.

Finally, Appendix A integrates the code in this documentation into a single listing for simulating scattering of a plane wave.

2 Preliminaries

This user guide describes the use of TMATROM3 to simulate scattering of a plane wave

$$\psi^{\mathrm{i}}(\boldsymbol{r}) = e^{ik\boldsymbol{r}\cdot\boldsymbol{d}}$$

with (unit vector) incident direction $\boldsymbol{d}$ and wavenumber k, where $k = 2\pi/\lambda$ and λ is the incident wavelength (see the accompanying paper [2] for a complete description of the wave scattering problem). The simulation process is visualised in the state-transition diagram Figure 1.

The incident wave is approximated by a truncated regular wavefunction expansion

$$\psi^{\mathrm{i}}(\boldsymbol{r}) = \sum_{l=0}^{n}\sum_{|m|\le l} a_{lm}\,\mathrm{Reg}\,\psi_{lm}(\boldsymbol{r}-\boldsymbol{r}_0)$$

about the local origin $\boldsymbol{r}_0$, with truncation parameter n. The scattered wave is approximated by a truncated radiating wavefunction expansion

$$\psi^{\mathrm{scat}}(\boldsymbol{r}) = \sum_{l=0}^{n}\sum_{|m|\le l} b_{lm}\psi_{lm}(\boldsymbol{r}-\boldsymbol{r}_0)$$

about the local origin $\boldsymbol{r}_0$. Here ψ_{lm} is the radiating wavefunction of degree l and order m, and $\mathrm{Reg}\,\psi_{lm}$ its regular part. We refer to Equations (15) and (4) in [2] for the truncation parameter and wavefunction definitions respectively.

It is convenient to begin by defining some key variables that we use in the examples below. We set the incident wave direction

```
    d = [0;0;1];
```

the wavenumber

```
    kwave = 3.1;
```

the local origin

```
    r0 = [0;0;0];
```

and the scatterer radius

```
    rad = 1.0;
```

Using the scatterer radius and wavenumber we choose the truncation parameter based on Equation (15) in [2] using

```
    n = suggestedorder(kwave,rad);
```

We will evaluate the incident field and the regular and radiating wavefunction expansions at various points in $\mathbb{R}^3$. In this manual we choose points on the intersection of the x-z plane and the unit sphere,

```
    theta = linspace(0,2*pi);
    r = [sin(theta);
      zeros(size(theta));
      cos(theta)];
```

Because these points are on the unit sphere, they can also be used for evaluating the far field.

3 Incident Wave

Setup incident wave

The incident plane wave is represented using

```
psi = plane_wave(d,kwave);
```

Evaluate

The incident wave ψ^{i} is evaluated at the points `r` using

```
f = psi.evaluate(r);
```

Evaluate gradient

The partial derivatives of the incident wave ψ^{i} with respect to x, y and z are calculated using

```
[dx,dy,dz] = psi.evaluateGradient(r);
```

4 Regular Expansions

Create regular expansion

The truncated regular wavefunction expansion is created using

```
a = regularwavefunctionexpansion(...
  n,r0,psi);
```

Evaluate

The expansion is evaluated at the points `r` using

```
u = a.evaluate(r);
```

Add and subtract

Expansions `a` and `b` that have the same origin and order can be added

```
a + b
```

and subtracted

```
a - b
```

Change origin

A new regular wavefunction expansion for `a` with local origin `r1` is created using

```
b = regularwavefunctionexpansion(...
  a,r1);
```

Rotate coordinate system

The coordinate system is rotated by angle `theta` about the y-axis using

```
a.rotatecoordinatesabouty(theta)
```

The coordinate system is rotated by angle `theta` about the z-axis using

```
a.rotatecoordinatesaboutz(theta)
```

5 Radiating Expansions

Create radiating expansion

The truncated radiating wavefunction expansion for the scattered field is obtained using

```
f = T * a;
```

where `T` represents the T-matrix.

Evaluate

The expansion is evaluated at the points `r` using

```
u = f.evaluate(r);
```

Evaluate far field

The far field of the expansion is evaluated at the points `r` using

```
u = f.evaluateFarField(r);
```

Add and subtract

Expansions `f` and `g` that have the same origin and order can be added

```
f + g
```

and subtracted

```
f - g
```

Change origin

A new radiating wavefunction expansion for `f` with local origin `r1` is created using

```
g = radiatingwavefunctionexpansion(...
  f,r1);
```

A new regular wavefunction expansion for `f` with local origin `r1` is created using

```
a = regularwavefunctionexpansion(...
  f,r1);
```

Note: the new regular wavefunction expansion is valid inside the ball of radius $\|\boldsymbol{r}_0 - \boldsymbol{r}_1\|_2$ with center $\boldsymbol{r}_1$.

Rotate coordinate system

The coordinate system is rotated by angle `theta` about the y-axis using

```
f.rotatecoordinatesabouty(theta)
```

The coordinate system is rotated by angle `theta` about the z-axis using

```
f.rotatecoordinatesaboutz(theta)
```

6 T-matrix

Use precomputed T-matrix

The T-matrix is created from a precomputed matrix M using

```
T = tmatrix(n,kwave,M,r0);
```

Here M must be an $(n+1)^2 \times (n+1)^2$ matrix.

Load T-matrix

A saved T-matrix is loaded using

```
T = tmatrix(filename);
```

Compute T-matrix

A T-matrix is computed using

```
T = ghtmatrix(n,kwave,slvr,r0);
```

where slvr is a solver object that has been setup. (see Section 7).

Estimate error

An estimate of the error in the T-matrix is obtained using

```
T.error()
```

This estimate, which is valid for non-absorbing scatterers, is based on how well the T-matrix satisfies the symmetry relation $T + T^* + 2TT^* = 0$ (see [2, Section I]).

Save

A T-matrix is saved using

```
T.save(filename)
```

7 Solvers

The solver is used to compute the T-matrix for a given scatterer. The choice of solver depends on the shape and material properties of the scatterer. TMATROM3 provides several solvers for ellipsoidal scatterers, an implementation of the method of fundamental solutions [1] for more general scatterers, and a template for users to define their own solvers.

We outline use of the solver for a sound-soft ellipsoid

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} + \frac{z^2}{r_z^2} = 1.$$

It is convenient to define the following variables that describe the scatterer

```
rx = 0.66;
ry = 0.66;
rz = 1.0;
```

and discretisation parameters

```
m = 25;
q = 2*m;
```

Setup solver

The solver is initialised using

```
slvr = solverEllipsoidSoft(...
  kwave,[],r0,[rx ry rz],m,q);
```

The solver approximates the scattered field using $2m^2$ discrete sources and matches the incident field at $2q^2$ points on the ellipsoid surface.

The solver is setup using

```
slvr.setup()
```

Solve

The incident field is specified using

```
slvr.setIncidentField(psi);
```

The associated scattering problem is solved using

```
slvr.solve()
```

Evaluate far field

The far field of the solution is evaluated using

```
slvr.getFarField(r);
```

References

[1] G. Fairweather and A. Karageorghis. The method of fundamental solutions for elliptic boundary value problems. *Adv. Comput. Math.*, 9:69–95, 1998.

[2] M. Ganesh and S. C. Hawkins. A numerically stable T-matrix method for acoustic scattering by nonspherical particles with large aspect ratios and size parameters. *J. Acoust. Soc. Am.*, 2022. to appear.

A Example listing

```
% setup preliminary variables
d = [0;0;1];
kwave = 3.1;
r0 = [0;0;0];
rad = 1.0;
rx = 0.66;
ry = 0.66;
rz = 1.0;
m = 25;
q = 2*m;
theta = linspace(0,2*pi);
```

```
r = [sin(theta);
  zeros(size(theta));
  cos(theta)];

% get expansion order
n = suggestedorder(kwave,rad);

% setup solver
slvr = solverEllipsoidSoft(...
  kwave,[],r0,[rx ry rz],m,q);
slvr.setup()

% Compute T-matrix
T = ghtmatrix(n,kwave,slvr,r0);

% estimate error in the T-matrix
T.error()

% setup incident wave
psi = plane_wave(d,kwave);

% create regular expansion
a = regularwavefunctionexpansion(...
  n,r0,psi);

% multiply by T-matrix to create
% radiating expansion
f = T * a;

% evaluate far field
u = f.evaluateFarField(r);

% plot cross section
semilogy(theta,abs(u).^2)
```

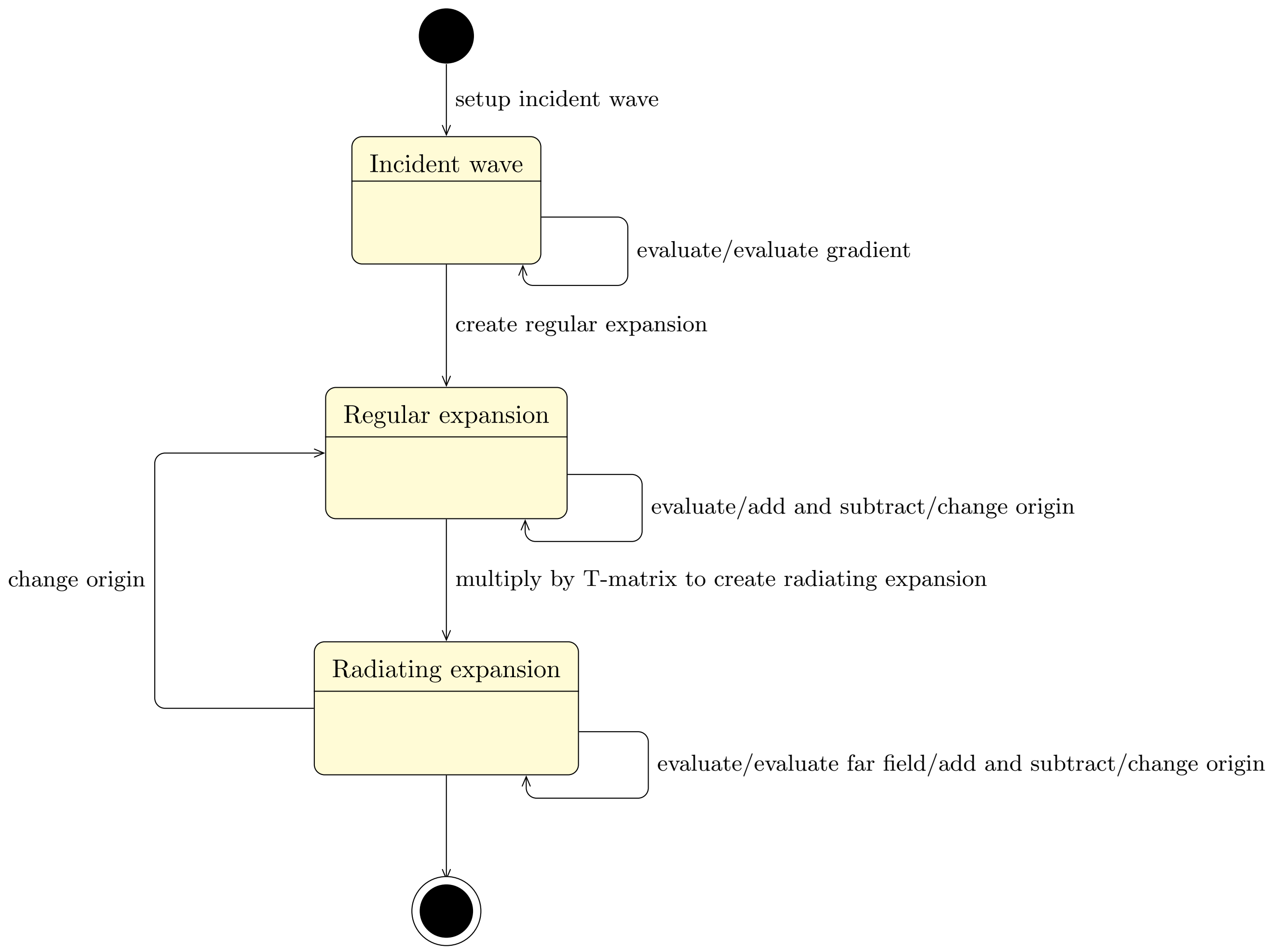
setup incident wave
Incident wave
evaluate/evaluate gradient
create regular expansion
Regular expansion
evaluate/add and subtract/change origin
change origin
multiply by T-matrix to create radiating expansion
Radiating expansion
evaluate/evaluate far field/add and subtract/change origin

Figure 1: State-transition diagram documenting the process for simulating scattering of a plane wave using TMATROM3.

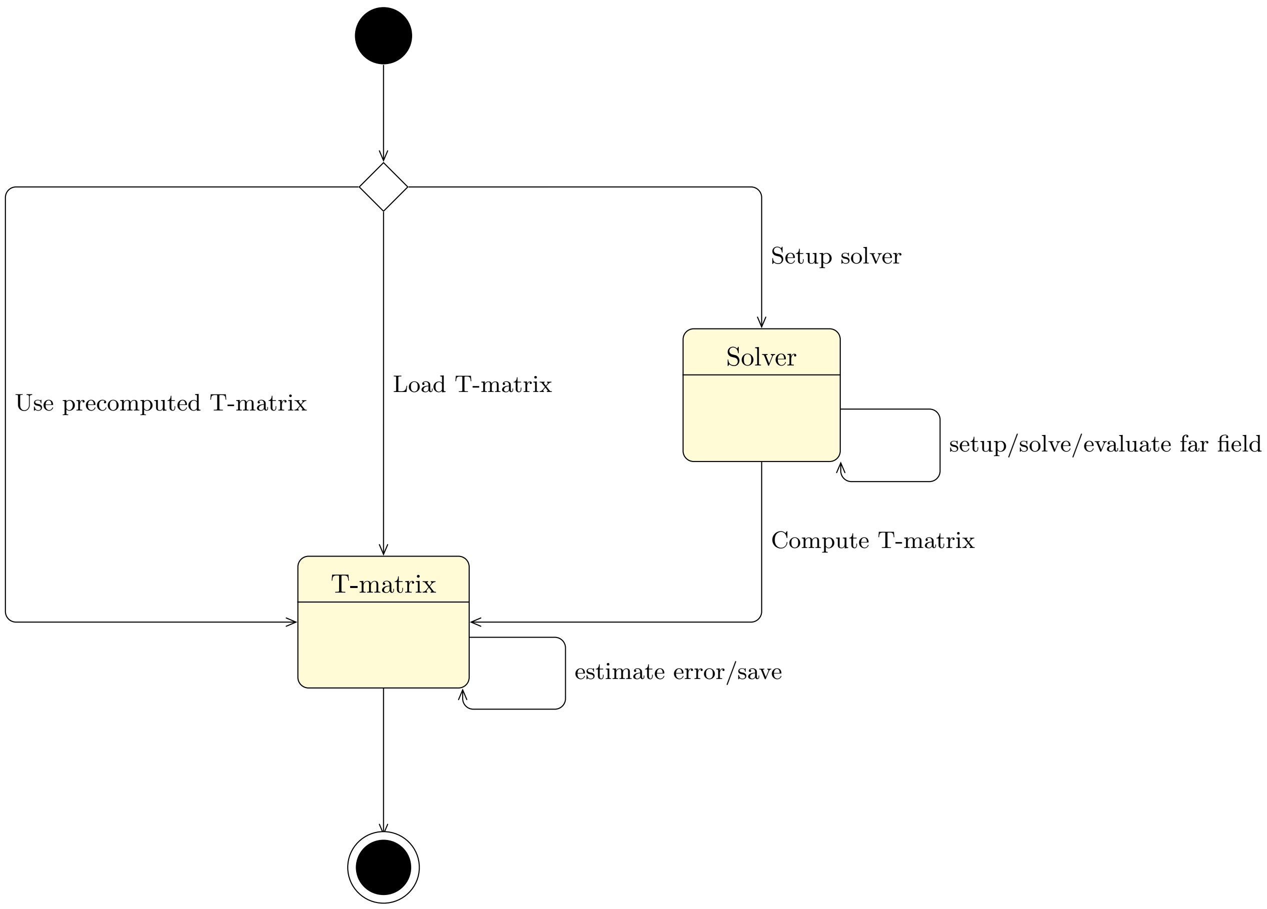

Figure 2: State-transition diagram documenting the process for setting up a T-matrix in TMATROM3.